AFRL-RH-WP-TR-2016-0005

Graphed-based Models for Data and Decision Making

Dr. Leslie Blaha

February 2016

Final Report

AIR FORCE RESEARCH LABORATORY
711 HUMAN PERFORMANCE WING,
HUMAN EFFECTIVENESS DIRECTORATE,
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC).

AFRL-RH-WP-TR-2016-0005 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//signed//
LESLIE M. BLAHA
Work Unit Manager
Battlespace Visualization Branch

//signed//
ROBERT C. MCKINLEY
Chief, Battlespace Visualization Branch
Warfighter Interface Division

//signed//
WILLIAM E. RUSSELL
Chief, Warfighter Interface Division
Human Effectiveness Directorate
711 Human Performance Wing

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
16-02-2016	Final	03 January 2012 – 31 December 2015

4. TITLE AND SUBTITLE
Graph-based Models for Data and Decision Making

5a. CONTRACT NUMBER
In-House

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Dr. Leslie M. Blaha

5d. PROJECT NUMBER
2313

5e. TASK NUMBER
CV

5f. WORK UNIT NUMBER
(H00B) 2313CV001

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
711HPW/RHCV
2255 H Street
Wright-Patterson AFB OH 45433-7022

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Materiel Command
Air Force Research Laboratory
711th Human Performance Wing
Human Effectiveness Directorate
Crew Systems Interface Division
Battlespace Visualization Branch
Wright-Patterson Air Force Base, OH 45433

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
711 HPW/RHCV

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RH-WP-TR-2016-0005

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution A: Approved for public release.

13. SUPPLEMENTARY NOTES
88 ABW Cleared 04/01/2016; 88ABW-2016-1588. Report contains color.

14. ABSTRACT

This effort is focused on developing and applying tools for modeling human information processing. Models include transformations of response time data from empirical studies, and complex network models for capturing broader dynamics of complex systems. Progress so far has resulted in some theoretical work in the area of response time modleing of workload capacity and dimer automata complex systems models of information transmission through a network. We have designed a novel approach for interfacing cogntive models and user interfaces, and have explored relationships between mathematical and computational models of a detection task. Additionally, new visual tools for pattern discovery and visual analytics are proposed based on topological data analysis theory. Several pieces of open source software have been developed for implementing these analyses and making them widely available.

15. SUBJECT TERMS
Workload capacity modeling, human information processing, complex networks, simplex, innovation diffusion, influence maximization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Leslie Blaha
Unclassified	Unclassified	Unclassified	SAR	105	19b. TELEPHONE NUMBER *(Include Area Code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Table of Contents

Section **Page**

1. Summary

1.1 Systems Factorial Technology with R

A portion of the effort to date has been dedicated to the development of an open source implementation of systems factorial technology (SFT) measures and models within the R for statistical computing framework and language. SFT is one methodology utilized in this research for making inferences about human information processing mechanisms utilizing response time data. The first version of the package (sft 0.1) was released in 2012; we published a tutorial paper on utilizing SFT, its associated experimental methodology, the double factorial paradigm, and the basic functionality in the sft package (Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R., & Townsend, J. T., 2013, Systems factorial technology with R. *Behavior Research Methods* [online publication doi 10.3758/s13428-013-0377-3). Additional research efforts have both contributed new theory to the SFT framework, but have continued to increase the functionality of the sft toolbox to include new measures. The second major release of the sft package (version 1.0-1) was made in November 2012, accompanied by a presentation of the new functions at the 2013 Society for Computers in Psychology Meeting. The latest update, 2.0-7 was released in October 2014. A companion tutorial paper on the new functions is currently under revision.

(Houpt, J. W., Blaha, L. M., & Burns, D. M. (under revision). Latest developments in systems factorial technology with R.)

1.2 Models of Opinion Dynamics

Dimer automata models provide a framework for modeling information dynamics of complex systems represented as networks. Several simulation studies were run exploring the ability of two- and three-state dimer automata systems to capture opinion dynamics (also termed innovation diffusion) and influence maximization in different networks. Simulation experiments examined different networks structures, the influence of zealotry on the dynamics, and strategies for the placement of zealots in the network for maximum influence on the final opinion states. Initial experiments were presented at the 2013 Behavior Representation in Modeling and Simulation conference, and additional experiments were included in an article published in 2015.

(Arendt, D. A. & Blaha, L. M., (2015) Opinions, influence and zealotry: A computational study on stubbornness. *Computational & Mathematical Organization Theory, 21*(2), 184-209 [invited paper]).

1.3 Generalized n-Channel Capacity Space

Theoretical progress was made in the area of parallel models of response time by the formulation of generalized bounds on the capacity coefficient values predicted by standard parallel processes with $n \geq 2$ channels in the system. Previously, general n-channel bounds (upper and lower) on the range of cumulative distribution functions for standard parallel models had been defined for minimum time, single-target self-terminating maximum time stopping rules. Relatedly, capacity coefficient ratios had been defined for the same three stopping rules. Because the capacity coefficients are formulated by logarithmic transformations of the cumulative distribution functions, we can redefine the bounds to provide upper and lower limits on the capacity coefficient functions directly. These capacity space bounds were derived and proven in an article published in 2015.

(Blaha, L. M. & Houpt, J. W. (2015). An extension of workload capacity space for systems with more than two channels. *Journal of Mathematical Psychology, 66,* 1-5.)

1.4 The Points to Pixels Pipeline ($P2P^2$)

In order for patterns to be found in and for meaningful information to be extracted from high dimensional or complex network data, easy to use and manipulate visualization tools are needed for data exploration. We developed an open

source framework for performing simplex clustering and visualizing data for visual analytics purposes. Data can be fed into the pipeline framework as either the raw multivariate measures, a (dis)similarity matrix computed from that data, or as a graph of network-type data. From any of those formats, the appropriate transformations of the data are made and then a simplex is derived. The parameters governing the computations are easily manipulated by the user. And a set of easy visualizations are created by fitting a convex hull to each clique or cluster in the data and projecting that into lower dimensional space, augmented by color coding. By utilizing a set of free, open source (Python based) toolboxes, the $P2P^2$ framework is easily utilized by any researchers without need for specialized software or expensive licensing.

(Arendt, D. L., Jefferson, B., & Su, S. (in preparation) The Points to Pixels Pipeline ($P2P^2$): and open source framework for multivariate, similarity, and network data visualization.)

1.5 SIMCog-JS: Simplified Interfacing for Modeling Cognition – JavaScript

Computational cognitive architectures have been limited in their application scope to post-experimental analysis or near real-time simulation separate from human operators. We propose that cognitive models could be used as real-time monitoring tools if they could execute a task simultaneously with the human operators. To enable this, we developed a client-server software architecture, Simplified Interfacing for Modeling Cognition – JavaScript (SIMCog-JS). This was implemented in JavaScript and employs websockets to enable communication between the Java ACT-R cognitive architecture and a JavaScript user interface. This software is available open source at http://sai.mindmodeling.org/simcog. A conference proceedings paper was published in 2015 at the International Conference on Cognitive Modeling.

(Halverson, T., Reynolds, B., & Blaha, L. (2015). SIMCog-JS: Simplified Interfacing for Modeling Cognition - JavaScript. *Proceedings of the International Conference on Cognitive Modeling*, Groningen, The Netherlands, April 9-11, 39-44.)

1.6 Modeling the Workload of Capacity of Visual Multitasking

We are extending the application of the capacity coefficient to multiple, simultaneously executed visual decision making tasks, which we refer to as visual multitasking. The initial testbed for this extension is an open-source JavaScript implementation of the modified Multi-Attribute Task Battery (mMATB; available online at http://sai.mindmodeling.org/mmatb). We presented initial results showing that two tasks can boost processing capacity, but moving to four tasks results in limited capacity processing on all tasks. Preliminary findings were published in the 2015 International Conference on Cognitive Modeling proceedings.

(Blaha, L. M., Cline, J., & Halverson, T. (2015). Modeling the workload capacity of visual multitasking. *Proceedings of the International Conference on Cognitive Modeling*, Groningen, The Netherlands, April 9-11, 37-38.)

1.7 ACT-R and LBA Model Mimicry Reveals Similarity Across Levels of Analysis

Computational and mathematical cognitive models have trade-offs in their implementations of explicit cognitive mechanisms and their mathematical tractability. This effort compared a mathematical sequential sampling model, the Linear Ballistic Accumulator (LBA), and a computational cognitive model programmed in the cognitive architecture Adaptive Control of Thought - Rational (ACT-R) of a simple detection task, the psychomotor vigilance task. We found that both provided good-fitting explanations of empirical data. Relationships were found between parameters. This enables future modeling to leverage the mathematical tractability of the LBA for fitting data and relationships between parameters to leverage the mechanistic explanations afforded by the computational cognitive architecture for understanding task behaviors in human observers. Areas for further theoretical development were identified. This work was published in a conference proceeding paper in 2015.

(Fisher, C. R., Walsh, M., Blaha, L. M., & Gunzelmann, G. (2015, July). ACT-R and LBA model mimicry reveals similarity across levels of analysis. *37th Annual Conference of the Cognitive Science Society*, Pasadena, California.)

1.8 Exploring Individual Differences via Clustering on Capacity Coefficients

Capacity coefficient analyses are performed at the individual participant level of data. In order to make inferences about groups or patterns within groups of participants, researchers have had to rely on visual inspection and qualitative descriptions. We present an approach for using functional principle components analysis to map all participants into a common vector space for machine learning analysis. We demonstrate how clustering can identify subgroups within the data that might relate to experimental manipulations or to participant population characteristics (age, diagnosis, etc.). This provides a set of tools for quantitative descriptions of individual differences in capacity coefficient data. Aspects of this technique were published in a 2015 conference proceedings paper; the full technique will appear in a book chapter in 2016.

(Houpt, J. W. & Blaha, L. M. (2015, July). Exploring individual differences via clustering on capacity coefficients. *37th Annual Conference of the Cognitive Science Society*, Pasadena, California.)

(Blaha, L. M. & Houpt, J. W. (2016, anticipated). Combining the capacity coefficient with statistical learning to explore individual differences. In *Mathematical Models of Perception and cognition: A Festschrift in honor of James T. Townsend* (J. W. Houpt & L. M. Blaha, Eds.). Psychology Press.)

2. Manuscripts from the Current Effort

Included in the following pages are drafts of manuscripts based on the efforts described above. Each of these are embedded images from a pdf document that was typeset in LaTeX. For documents 1.5-1.8, clicking on the pdf embedded in this document will open the pdf.

Latest Developments in Systems Factorial Technology with R

Joseph W. Houpt

Wright State University, Dayton, Ohio

Leslie M. Blaha

U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, Ohio

Devin M. Burns

Indiana University, Bloomington, Indiana

Author Note

Joseph W. Houpt

Department of Psychology

Wright State University

3640 Colonel Glenn Highway

Dayton, Ohio 45435

joseph.houpt@wright.edu

This work was supported by AFOSR grants FA9550-13-1-0087 to J.W.H. and LRIR 12RH14COR to L.M.B.

Abstract

Systems factorial technology (SFT) is a powerful and mathematically rigorous framework for studying how cognitive systems make use of multiple sources of information. Articles about SFT tend to focus on the mathematics and development of the theory, making them inaccessible to many researchers. The **sft** package for R was recently introduced to facilitate the use of SFT by a wider range of researchers. The original package contained tools implementing only the basic theoretical tools. In the last few years, there have been a number of advances to SFT, which we will review, and we introduce their implementation in the **sft** package. In particular, we will demonstrate R functions for functional principal components analysis of the capacity coefficient (Burns, Houpt, Townsend, & Endres, 2013), calculating and plotting assessment functions (Townsend & Altieri, 2012), and calculating and plotting distributional bounds in a unified capacity space (Townsend & Eidels, 2011). Additionally, we expanded the package to include a function for the new capacity coefficient for single-target self-terminating (ST-ST) processing (Blaha, 2010), as well as functions supporting the plotting of cumulative distribution function bounds on the predictions of standard parallel processing models for minimum time, maximum time, and ST-ST decision rules.

Latest Developments in Systems Factorial Technology with R

Introduction

Systems Factorial Technology (SFT) is a framework for analyzing how multiple sources of information are used together in cognitive processing. Although the tools are quite powerful and broadly applicable, they can be inaccessible, or at least daunting, to psychology researchers. Houpt, Blaha, McIntire, Havig, and Townsend (2013) introduced an R (R Development Core Team, 2011) package to implement the basic measures and statistical analyses. However, SFT continues to advance and more tools continue to become available. In this article we give an overview of the new theoretical advancements in the SFT framework and describe their use and implementation in the **sft** R package. In particular we focus on four advances from the last few years: the single-target self-terminating (ST-ST) capacity coefficient (Blaha, 2010; Blaha & Townsend, under review), the unified workload capacity space measures (Townsend & Eidels, 2011), functional principal components analysis (fPCA) of the capacity coefficient (Burns et al., 2013), and the workload assessment functions (Townsend & Altieri, 2012).

We will begin with an overview of workload capacity in SFT to give readers who may be less familiar with the topic a foundation for the rest of the paper. This overview is brief and meant only to give readers the basic details needed to use these new analyses. We encourage readers wanting further details to read the SFT with R paper (Houpt et al., 2013) or some of the original papers on workload capacity in SFT and on the capacity coefficient (Townsend, 1974; Townsend & Ashby, 1983; Townsend & Nozawa, 1995; Townsend & Wenger, 2004; Wenger & Townsend, 2000).

First, a brief note on our notation. When we refer to the R package for the implementation of SFT theory, we will use **sf**t. Any R code itself, like function names or input arguments, will be typeset as follows: `function` or `input.argument=value`.

Workload Capacity and the Capacity Coefficient

Within SFT, workload capacity refers to a change in information processing performance as the number of information sources change. The original definitions focused on processing speed as measured by response times. Some of the recent generalizations discussed in this paper and implemented in the latest version of the R package include response accuracy as well. In this section we will focus on the response time only approach, then discuss the generalization in the Assessment Function section.

In most cases, a system takes longer to finish the more it has to do. However, just because a system takes longer to respond when it is required to process more sources of information, it does not mean that any of the individual information sources are processing at a slower speed. Likewise, when there is redundant information available, the overall processing speed being faster does not mean that the processing of any individual source is faster. For example, in parallel processes with redundant information, faster processing times may be due to statistical facilitation (Raab, 1962; Miller, 1982). Statistical facilitation refers to the fact that the minimum over a set of more than one random variable (i.e., source processing times) tends to be smaller than any of the individual random variables. Statistical inhibition refers to the analogous phenomenon when all processes must finish: the maximum of multiple random variables tends to be larger than any of the individual random variables. Thus, if all we can measure is a person's response time with one or more sources of information present, and not the individual processing times of each source of information when multiple sources are available, it is important to compare the times against an appropriate baseline.

The baseline for the capacity coefficient in redundant target tasks is the unlimited-capacity, independent, parallel, first-terminating model (Townsend & Nozawa, 1995). We use the initialism UCIP for the first three assumptions and OR to refer to first-terminating (in reference to a logical OR decision rule). Because it is first-terminating, the model is finished as soon as any of the individual target processes have completed.

Equivalently, the model has not yet finished only if none of the individual target processes have finished,

$$\Pr\{T_{\text{UCIP-OR}} > t\} = \Pr\{T_1 > t, \ldots, T_n > t\}.$$

We can use T_i to refer to the processing time for the ith target regardless of whether there are other sources present due to the unlimited capacity assumption. Using the independence assumption, we can split the right side into a product,

$$\Pr\{T_1 > t, \ldots, T_n > t\} = \Pr\{T_1 > t\} \times \cdots \times \Pr\{T_n > t\}.$$

We can rewrite this equality more succinctly using survivor functions,
$S(t) = 1 - F(t) = \Pr\{T > t\}$,

$$S_{\text{UCIP-OR}}(t) = S_1(t) \times \cdots \times S_n(t)$$

where $F(t) = \Pr\{T \leq t\}$ is the cumulative distribution function. Lower survivor functions correspond to faster processing times. To translate this identity to cumulative hazard functions we use $H(t) = -\log S(t)$, so we see that larger cumulative hazard functions correspond to faster processing times.

The cumulative of the hazard function is convenient for statistical purposes and has the nice interpretation as the amount of work completed by the cognitive processing system in t amount of time. We take the natural logarithm of both sides of the previous equation to arrive at the baseline prediction of the UCIP-OR model in terms of cumulative hazard functions,

$$H_{\text{UCIP-OR}}(t) = H_1(t) + \cdots + H_n(t).$$

The capacity coefficient is a ratio function comparing this UCIP model baseline to observed performance. Let $C = \{1, \ldots, n\}$ denote the set of n active channels in an experiment. Using this set notation, we denote the empirical response time cumulative distribution function (CDF) on an OR task as $F_C(t) = P\left[\min_C(T_c) \leq t\right]$, for all real $t \geq 0$ and $c \in C$. The corresponding empirical cumulative hazard function is denoted $H_C(t)$. The

capacity coefficient for tasks in which a first-terminating decision rule is expected is given by the ratio of cumulative hazard functions of response times when all n targets are present to the sum of cumulative hazard functions of response times for cases when each of the n targets is present in isolation,

$$C_{\text{OR}}(t) = \frac{H_C(t)}{H_{\text{UCIP-OR}}(t)} = \frac{H_C(t)}{H_1(t) + \cdots + H_n(t)}. \tag{1}$$

The baseline of UCIP-OR processing is estimated in the denominator, so if the performance measured when all targets sources are present is better than the estimated baseline, then $C_{\text{OR}}(t) > 1$. Likewise, worse than baseline performance would be indicated by $C_{\text{OR}}(t) < 1$.

The same logic can be used to derive the baseline for tasks in which the participant can only respond when all sources of information have been processed, i.e. exhaustive or AND tasks. For the UCIP-AND model to finish, it must finish processing all sources of information,

$$\begin{aligned} \Pr\{T_{\text{UCIP-AND}} \leq t\} &= \Pr\{T_1 \leq t, \ldots, T_n \leq t\} \\ F_{\text{UCIP-AND}}(t) &= F_1(t) \times \cdots \times F_n(t). \end{aligned}$$

In terms of the cumulative reverse hazard function, $K(t) = \log F(t)$,

$$K_{\text{UCIP-AND}}(t) = K_1(t) + \cdots + K_n(t).$$

Lower CDFs correspond to slower processing, so lower cumulative reverse hazard functions correspond to worse performance. Because $F(t)$ is between 0 and 1, the logarithm of $F(t)$ is always negative, so lower values correspond to larger magnitudes. Hence, to keep the interpretation of $C(t) > 1$ corresponding to better than baseline, the AND capacity coefficient is flipped,

$$C_{\text{AND}}(t) = \frac{K_1(t) + \cdots + K_n(t)}{K_C(t)}. \tag{2}$$

Note that for the observed performance in an AND task, we use the response time CDF $F_C(t) = P\left[\max_C(T_c) \leq t\right]$, for all real $t \geq 0$ and $c \in C$, and we denote the cumulative

reverse hazard function $K_C(t)$. The baseline is now represented in the numerator, so larger magnitude cumulative reverse hazard functions for response times to all sources of information (the denominator) indicates worse than baseline performance and leads to $C_{\text{AND}}(t) < 1$. Likewise, better performance than the baseline leads to $C_{\text{AND}}(t) > 1$.

Experimentally, workload capacity analysis can be used on any tasks that require an AND or OR type of decision (and now single-target self-terminating, as we will explain below) and that utilize a manipulation that involves judgments on different numbers of information sources. There are two specific workload manipulations needed to utilize Equations 1 and 2. The first is a set of single information source trials that allow the estimation of the individual channel response time distributions. This is required for the UCIP baseline model estimates. The second necessary condition is one in which all the sources of information are presented together, to estimate the actual cognitive processing of n active channels. For more on the experimental manipulations for capacity analysis, particularly in the context of the double factorial paradigm, see Houpt et al. (2013).

To make this concrete, imagine a visual or memory search task. In order to estimate the UCIP baseline model, participants must complete a series of single-target trials (i.e. one item in the search array) with one type of trial for each individual different source of information. Participants must also complete trials for n items in the search array. If this array was all targets, then participants would be completing an OR redundant-targets task, and the experimenter would use Equation 1 for his analysis. If this array was all distractors, and all must be searched to determine the target was not present, then participants would be completing an AND task, and the experiment would use Equation 2 for the analysis for those response times.

Functions for calculating the traditional capacity coefficients and the associated test statistics from (Houpt & Townsend, 2012) are available in the **sft** package and described in Houpt et al. (2013). With the basics of workload capacity analysis in SFT established, we can now summarize the latest developments and their corresponding functions in the **sft**

package.

Single-Target Self-Terminating Capacity

Single-target self-terminating (ST-ST) processing refers to a response rule that sits between OR and AND processing. This is the condition where there is a single target of interest for the response. When this target is presented among other non-target information sources in a task, it may be the first or last item processed or somewhere in between. However, as soon as the target is identified, the observer can make a response (hence, the nomenclature 'self-terminating'). For example, ST-ST processing is often the stopping rule demanded in a visual or memory search task when a single target of interest is embedded in a search array of distractors.

As with AND processing, the ST-ST capacity coefficient compares performance on a task to a UCIP model using cumulative reverse hazard functions (Blaha, 2010; Blaha & Townsend, under review). The UCIP model prediction is the cumulative reverse hazard function for response times to the single target processed in isolation. Let $K_k(t)$ denote the response time cumulative reverse hazard function for single-target k processed alone. Because the assumptions of the UCIP model are that the individual channel processing rates are independent of other channels and do not change as the total number of channels changes, then

$$K_{\text{UCIP-STST}} = K_k(t).$$

The cumulative reverse hazard function for processing of the same single target k among n total information sources ($n-1$ distractors) is denoted $K_{k,C}(t)$, where again $C = \{1, \ldots, n\}$. The latter case is the higher workload condition of interest for workload capacity analysis. Taking a ratio of the UCIP model to the n-source processing performance gives the ST-ST capacity coefficient:

$$C_{\text{STST}}(t) = \frac{K_k(t)}{K_{k,C}(t)}. \tag{3}$$

Similar to $C_{\text{AND}}(t)$, the numerator is the baseline model, and a larger denominator indicates worse than baseline performance, giving $C_{\text{STST}}(t) < 1$, which is referred to as limited capacity processing. This indicates that either there are limited processing resources available, there is inhibition among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed serially).

Likewise, better than baseline performance again leads to $C_{\text{STST}}(t) > 1$, which is referred to as super capacity processing. This indicates that either there are more processing resources available per process when there are more processes, that there is facilitation among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed coactively).

Additionally, Blaha and Townsend (under review) showed that a statistical test for $C_{\text{STST}}(t)$ is a special case of the statistical test for AND capacity developed by Houpt and Townsend (2012). The estimator of the cumulative reverse hazard function is calculated with the `estimateNAK` function in the **sft** package, as covered in Houpt et al. (2013).

In the **sft** R package, the ST-ST capacity coefficient and corresponding statistical test (Blaha & Townsend, under review) are calculated by the `capacity.stst` function. It takes as its input a list containing two arrays of response time data. The first array in the list is assumed to be the response times from the single-target self-terminating condition with a total of n information sources, and the second array in the list is assumed to be the response times from the single target processed in isolation (the baseline estimate). The second input argument is an optional list of arrays of correct indicators; if the correct indicators are not provided (`CR=NULL`), the function assumes that all response times are from correct responses.

Finally, the `capacity.stst` function includes an indicator input `ratio`. If `ratio=TRUE`, then the ratio form of the capacity coefficient (Equation 3) is returned; examples of ratio $C_{\text{STST}}(t)$ functions, simulated for super capacity, unlimited capacity, and limited capacity models, are shown in Figure 1. If `ratio=FALSE`, then the difference form

of the capacity coefficient is returned. The difference form of the ST-ST capacity coefficient is given by

$$C_{\text{STST}}(t) = K_{k,C}(t) - K_k(t). \tag{4}$$

For the difference form of $C_{\text{STST}}(t)$, the reference value for unlimited capacity processing is 0 instead of 1. Negative values indicate worse than UCIP performance, and positive values indicate better than UCIP performance.

We can start with an simulated example data set to demonstrate the `capacity.stst` function. Recall that we need two sets of response times, the single target in isolation and the single target among other non-target processes. In this example, we simulate data from a limited-capacity condition, wherein the additional information sources slowed the processing rate of our target channel,

```
rate1 <- .35
RT.pa <- rexp(100, rate1)
RT.pp.limited <- rexp(100, .5*rate1)
tvec <- sort(unique(c(RT.pa, RT.pp.limited)))
```

To evaluate $C_{\text{STST}}(t)$ and test the null hypothesis of UCIP-STST processing, we can use the function with a list of response time vectors.

```
cap <- capacity.stst(RT=list(RT.pp.limited, RT.pa))
```

We use `print(cap$Ctest)` to see the results of the statistical test.

```
Houpt-Townsend UCIP test
data:  RT and CR
z = -3.4161, p-value = 0.0006353
alternative hypothesis: response times are different than those
predicted by the UCIP-AND model
```

The z-score is significantly negative, so we would reject the null hypothesis of UCIP-STST processing. Note that in this example, we used the default function calls of

`CR=NULL` (i.e., we assume all response times are from correct trials) and `ratio=TRUE` (return the ratio version of the function). Also, note that the information about the alternative hypothesis returned with the `print` command refers to the UCIP-AND model, because the statistical test is a special case of the AND test with only a single channel in the UCIP model (c.f. Blaha and Townsend (under review)). The data from this simulated example are plotted as the solid red line in Figure 1.

The `capacity.stst` function returns an `approxfun` object representing the ST-ST capacity ratio function (`ratio=TRUE`, which is the default) or the ST-ST capacity difference function (`ratio=FALSE`), as well as the `ucip.test` for ST-ST processing. If `ratio=FALSE`, `capacity.stst` also returns the variance estimate for the difference variant for the capacity coefficient. If the reported p-value for the statistical test is less than the user's predetermined type I error α level, at least one of the UCIP assumptions has failed.

Unified Workload Capacity Space

Townsend and Eidels (2011) introduced unified capacity spaces, a set of inequalities that enable both capacity coefficients and the parallel processing response time distribution bounds to be plotted on the same coordinate system for direct visual comparison. In order to do this, the bounds for standard parallel processing were transformed from standard CDF values existing on the range $[0, 1]$ to inequalities of either cumulative hazard functions or cumulative reverse hazard functions, depending on the stopping rule, for direct comparison with the capacity coefficient values. Note that in this case, the capacity coefficient assumes the ratio format which exists on the range $[0, +\infty]$. Townsend and Eidels (2011) derived the unified capacity space inequalities for AND and OR processing of 2-channel systems. (Blaha & Houpt, Under Review) extended this theory to general n-channel models and derived the unified space inequalities for ST-ST processing.

In the **sft** package, we have developed a single function, `estimate.bounds`, that can estimate both the traditional CDF versions of the bounds on parallel processing for all

stopping rules and the unified workload capacity space inequalities. First we review both versions of the inequalities, and then we explain the `estimate.bounds` function.

Bounds on Standard Parallel Processing

OR. Let $F_C(t) = P\left[\min_C(T_c) \leq t\right]$, for all real $t \geq 0$ and $c \in C$, denote the cumulative distribution of response times under a minimum time (logical OR) stopping rule. The general bounds for n-channel parallel processing under an OR stopping rule are (Colonius & Vorberg, 1994):

$$\max_i \left[F_{C\setminus\{i\}}(t)\right] \leq F_C(t) \leq \min_{i,j} \left[F_{C\setminus\{i\}}(t) + F_{C\setminus\{j\}}(t) - F_{C\setminus\{i,j\}}(t)\right]. \tag{5}$$

Here, we have used the set notation $C \setminus \{i\}$ to indicate reponse times with all sources present except i (i.e. $n-1$ total processing channels). Under the assumption or conditions that the individual channels are identically distributed (IID), this inequality chain simplifies to

$$F_{C\setminus\{1\}}(t) \leq F_C(t) \leq \left[2 * F_{C\setminus\{1\}}(t) - F_{C\setminus\{1,2\}}(t)\right]. \tag{6}$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$\min\left[F_1(t), F_2(t)\right] \leq F_{\{1,2\}}(t) \leq \left[F_1(t) + F_2(t)\right]. \tag{7}$$

The upper bound on this final inequality is often referred to as the 'race-model inequality,' which has long been used to test for evidence of coactive processing architecture (Miller, 1982).

AND. Let $G_C(t) = P\left[max_C(T_c) \leq t\right]$, where again $C = \{1, \ldots, n\}$ is the set of all n channels and $c \in C$, denote the cumulative distribution of response times under a maximum time (logical AND, exhaustive) stopping rule. The general bounds for n-channel parallel processing under an AND stopping rule are (Colonius & Vorberg, 1994):

$$\max_{i,j} \left[G_{C\setminus\{i\}}(t) + G_{C\setminus\{j\}}(t) - G_{C\setminus\{i,j\}}(t)\right] \leq G_C(t) \leq \min_i \left[G_{C\setminus\{i\}}(t)\right]. \tag{8}$$

Under the assumption or conditions that the individual channels are identically distributed, this inequality chain simplifies to

$$\left[2 * G_{C\setminus\{1\}}(t) - G_{C\setminus\{1,2\}}(t)\right] \leq G_C(t) \leq G_{C\setminus\{1\}}t). \tag{9}$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$[G_1(t) + G_2(t) - 1] \leq G_{\{1,2\}}(t) \leq \min\left[G_1(t), G_2(t)\right]. \tag{10}$$

ST-ST. Let $F_{k,C}(t) = P[T_{k,C} \leq t]$ denote the CDF of response times under the ST-ST stopping rule, where the target of interest is on processing channel k among n active channels. The general bounds for n-channel parallel processing under an ST-ST stopping rule are (Blaha & Townsend, under review):

$$\prod_{c=1}^{n} F_c(t) \leq F_{k,C}(t) \leq \sum_{c=1}^{n} F_c(t). \tag{11}$$

Under the assumption or conditions that the individual channels are identically distributed, this inequality chain simplifies, for any channel $c \in C$, to

$$[F_c(t)]^n \leq F_{k,C}(t) \leq n * F_c(t). \tag{12}$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$[F_1(t) * F_2(t)] \leq F_{k,\{1,2\}}(t) \leq [F_1(t) + F_2(t)]. \tag{13}$$

Note that in this case, $k = 1$ or $k = 2$, but this may not be specifiable *a priori* depending on experimental design.

Across all stopping rule conditions, violation of the upper bound indicates performance that is faster than can be predicted by an unlimited capacity parallel model. This may arise from positive (facilitatory) crosstalk between parallel channels, super capacity parallel processing, or some form of co-active architecture in the measured human response time data. Violation of the lower bound indicates performance that is slower than

predicted by an unlimited capacity parallel model. This may arise from negative (inhibitory) crosstalk between parallel channels, fixed or limited capacity processing, or some form of serial architecture in the measured human response time data.

Bounds on Capacity Coefficient Space

The bounds on parallel processing defined above can be transformed from CDFs into cumulative hazard and cumulative reverse hazard functions to form inequality chains with the capacity coefficients. The bounds for all stopping rules and all models are summarized in Table 1. For the derivation of these bounds, the reader is referred to Townsend and Eidels (2011) and Blaha and Houpt (Under Review).

The `estimate.bounds` function in the **sft** package can be flexibly used to compute either the CDF or unified capacity space bounds on standard parallel processing. For its first input argument, `RT`, it takes a list of numeric arrays of response times, each measured from the individual channels to be modeled. The `RT` list can contain either one array for each of the n channels to be estimated (so `length(RT)=n`), or it can have `length(RT)=1` and the bounds can be found under an assumption that the n channels are identically distributed. In the former case, the number of channels, n, is estimated from the length of the `RT` list, and so the user can keep the default input arguments `assume.ID=FALSE` and `numchannels=NULL`. In the latter case, because the length of the `RT` list is only 1, the input arguments `assume.ID=TRUE` and `numchannels=n` (where $n \geq 2$) must be specified by the user.

The optional input argument `CR` is a list of correct indicators that should have the same length as the input argument `RT`. If `CR=NULL` (default), then all the response times are assumed to be from correct response trials.

Critically, the user must specify which stopping rule (OR, AND, ST-ST) should be computed using the argument `stopping.rule=("or", "and", "stst")`. Finally, the input argument `unified.space` indicates whether the bounds should be computed for

CDF space (`unified.space=FALSE`) or for the unified capacity coefficient space (`unified.space=TRUE`).

Here, we demonstrate the use of the `estimate.bounds` function with data from the `dots` dataset, which is included with the **sft** package. First, we load the data and extract the necessary data to estimate the bounds for Participant S3 for the OR stopping rule condition.

```
data(dots)
attach(dots)
sub <- 'S3'
cond <- 'OR'
chan1 <- RT[Subject==sub & Condition==cond & Correct & Channel1>0 & Channel2==0]
chan2 <- RT[Subject==sub & Condition==cond & Correct & Channel1==0 & Channel2>0]
redundant <- RT[Subject==sub & Condition==cond & Correct & Channel1>0 & Channel2>0]
rts <- list(redundant,chan1, chan2)
```

Next, we calculate the bounds using the `estimate.bounds` function.

```
cdf.bounds <- estimate.bounds(rts[2:3], corrects[2:3], stopping.rule='or')
capacity.bounds <- estimate.bounds(rts[2:3], corrects[2:3],
stopping.rule='or', unified.space=TRUE)
```

We then calculate the redundant targets cdf to compare to bounds.

```
redundant.cdf <- ecdf(rts[[1]][corrects[[1]]>0])
```

And, we calculate the capacity coefficient.

```
or.cap <- capacity.or(rts, corrects)
```

Sample plots of parallel processing bounds computed with `estimate.bounds` are shown in Figure 2. This figure shows both the AND and OR bounds, plotted in both CDF and unified capacity space, for a single participant from the `dots` data set. In the CDF space plots, the empirical CDF of the redundant target trials response time data for either

the AND and OR conditions is shown in the thick, solid black lines. The upper and lower bounds on those CDFs are plotted in the dashed and dotted (respectively) red lines. Note that in these traditional views, we would try to make inferences about capacity from the violations of the bounds.[1] For example, in the data shown in Figure 2 (lower half, OR task), there is a clear violation of the lower bound, roughly between 0 and 250 ms. Using the traditional CDF space plots, we would infer that Participant S3 is too slow to be performing like a race model with redundant targets. Now, using the unified capacity space plots, we can make more direct inferences about the relationships of the bounds and capacity coefficient. In the lower right plot of Figure 2, limited capacity $C_{\mathrm{OR}}(t) < 1$ is observed for the whole range of response times, with violations of the lower bound obvious for the early response times.

fPCA for Capacity Coefficients

Functional principal components analysis (fPCA) is an extension of standard principal components analysis to infinite dimensional (function) spaces (c.f. Ramsay & Silverman, 2005). Just as in standard principal components analysis, fPCA is a method for finding a basis set of lower dimensionality than the original space to represent the data. However, in place of basis vectors, fPCA has basis functions. Each function in the original dataset can then be represented by a linear combination of those bases, so that each datum is represented by a vector of coefficients (or scores) in that linear combination.

The capacity coefficient is a function across time, so the differences among capacity coefficients from different participants and/or conditions can rarely be characterized by simple greater than or less than relations. The nuances of variation in functions would be lost if one were to reduce the capacity estimates to a point by taking an average across time or the maximum/minimum of the function. By using fPCA we can maximize the

[1] For a full discussion of the inequality chains formed by the AND and OR processing bounds, as well as the inferences about capacity that are possible from these inequality chains, the reader is referred to Townsend and Wenger (2004).

amount of variation we capture with a point estimate or small number of values: The factor scores can be used to examine differences among capacity coefficients, taking into account variation across the entire function.

The R function for fPCA implements the steps outlined in Burns et al. (2013). First, the data are shifted by subtracting the median response time within each condition for each participant, using the same shift for both single target and multiple target trials, so that the capacity curves will be registered. Second, each capacity coefficient is calculated with the shifted response times. Next, the mean capacity coefficient across participants and conditions is subtracted from each capacity coefficient, and the resulting capacity coefficients are represented using a b-spline basis. The fPCA procedure extracts the first basis function from the bspline space that accounts for the largest variation across the capacity coefficients. The next basis function is chosen as that which explains the largest amount of remaining variation in the capacity coefficients, given the constraint that it must be orthogonal to the first. This process continues until the indicated number of bases have been extracted.[2] Once the capacity functions are represented in the reduced space, a varimax rotation is applied to concentrate variability and increase interpretability.

The `fPCAcapacity` function can be called from the **sft** package using the following syntax:

```
fPCAcapacity(sftData, dimensions, acc.cutoff = .75, OR = TRUE, ratio = TRUE,
plotPCs = FALSE)
```

The data for fPCA analysis should be in the standard SFT data form, which is described thoroughly in Houpt et al. (2013): there should be a column for a participant identifier (`sftData$Subject`), a column for the condition (`sftData$Condition`), a column for the salience manipulation value of each source of information (`sftData$Channel`i), a column for response times (`sftData$RT`), and finally a column indicating whether the participant was correct on each trial (`sftData$Correct`). The `fCPAcapacity` function also has a

[2] The maximum possible number of basis functions is the number of input functions.

`ratio` flag to indicate whether to output capacity ratios (if `ratio=TRUE`) or differences, an `OR` flag indicating the version of the capacity coefficient (Equation 1 if `OR=TRUE`; Equation 2 if `OR=FALSE`),[3] and an `acc.cutoff` input value to establish a minimum criterion for accuracy required for including data in the analysis. Two variables unique to the fPCA analysis are the `dimensions` value, which can be set by the experimenter to establish the number of basis functions used to represent the data, and the `plotPCs` indicator which will generate plots of the principal components if `plotPCs=TRUE`.

The output of the function is a list of length four. The first list entry is a data frame titled `Scores`, which contains the loading values (coefficients on the basis functions) for each participant and condition. `MeanCT` is the averaged capacity function across all participants and conditions, while `PF` is a list containing each of the principal functions, the number of which will have been specified by the dimensions argument in the call to the function. The last list entry is `medianRT`, which will keep track of the amount each capacity curve has been shifted during the registration step, measured in milliseconds of RT.

Figure 3 illustrates the output plots generated by the `fPCAcapacity` function when run on the `dots` data using the function call:

```
fPCAcapacity(dots, 2, acc.cutoff = .75, OR = TRUE, ratio = TRUE,
plotPCs = TRUE).
```

Note that in the `dots` data, there are two conditions, OR and AND, referring to two task instructions given in the experiment; in the present analysis, we use Equation 1 in the fPCA analysis for all the data. In the above call, we asked for two dimensions, but again that choice is up to the experimenter. We can see that for the `dots` data, the first two components can together account for 93% of the variance (summing the values noted on the y-axis labels). The first component function mainly inflates (or deflates, depending on the sign of the loading value) capacity values for early- to mid-range reaction times. The second PC captures variation in the capacity function at early and late times; when PC2 is

[3]Note that the ST-ST capacity coeffient has not yet been implemented in `fPCAcapacity`.

higher, both early and late values of C(t) are higher. The scores for each of the ten participants, in the two stopping rule conditions, are shown in the right panel of Figure 3. In this example, both of the components can easily separate differences in the two tasks and between the various subjects. Combining the information from the Component plots and the Score values, the OR condition data are consistently higher than the AND condition data for all times and all participants. Within participants between conditions, the largest differences in capacity coefficient functions occur in the middle range of response times. fPCA also highlights differences in capacity among participants. In particular, participant S5 shows much lower variability between the OR and AND conditions than the other participants, and so S5's loading scores are higher and closer together in the right-hand plots.

Because the principal component functions are specifically chosen to describe the variability between the capacity functions for participants and conditions, this tool provides an excellent method for looking for influences of task and individual differences in capacity functions. Whereas most previous analyses of capacity data have restricted themselves to a gross comparison with the baseline model (i.e. observed value relative to 1), this analysis is more relative, highlighting differences between observed functions, and picking up dynamic patterns across various reaction times.

For more details on fPCA for the capacity coefficient, see Burns et al. (2013). For more general details on using fPCA in R, see Ramsay, Hooker, and Graves (2009).

Assessment Functions

The assessment functions are a generalization of the workload capacity functions that account for incorrect responses. The original capacity coefficient established a baseline that assumed perfect accuracy. While the standard capacity coefficient is robust to slightly less than perfect performance by a participant (the rule of thumb is that above roughly 90% accuracy should be fine), when accuracy is low, either the assessment functions or a

parametric measure such as the linear ballistic accumulator (LBA) capacity (Eidels, Donkin, Brown, & Heathcote, 2010) should be used.

Townsend and Altieri (2012) derived four different assessment functions each for AND and OR tasks to compare performance on two target information sources with the performance of an unlimited-capacity, independent, parallel (UCIP) model. The UCIP model is augmented with an error generating process for both sources of information. Each error process is assumed to be independent of, and parallel to, the processes for the other source of information, but there is no assumption of independence between the correct and error processes for the same source of information.

The correct assessment functions assess performance on correct trials and the incorrect assessment functions assess performance on the trials with incorrect responses. The fast assessment functions use the cumulative distribution functions, similar to the AND capacity coefficient, and the slow assessment functions use the survivor functions, similar to the OR capacity coefficient.

In an OR task, the detection model assumes that the response will be correct if it is correct on either source, i.e., if either source is detected. Hence, the first source (A) correct processing time must the faster than first source incorrect time, $T_{AC} < T_{AI}$ or the second source (B) correct must be faster than the second source incorrect, $T_{BC} < T_{BI}$. For the CDF (fast) version of the assessment function, we are interested in whether the response was at or before t, so either $T_{AC} \leq t$ and $T_{AC} < T_{AI}$ or $T_{BC} \leq t$ and $T_{BC} < T_{BI}$. Using f_{AC} for the completion time density for the first source correct process, F_{AI} for the distribution of first source, incorrect processes completion times, and likewise for the second source, this probability can be written out as,

$$\int_0^t f_{AC}(t)\left[1 - F_{AI}\right] + \int_0^t f_{BC}(t)\left[1 - F_{BI}\right] - \int_0^t f_{AC}(t)\left[1 - F_{AI}\right] \int_0^t f_{BC}(t)\left[1 - F_{BI}\right].$$

The same pattern of logic can be used to determine the baseline of processing for each of

other cases, slow-correct, fast-incorrect and slow-incorrect. For a full explication of the assessment functions and the derivation of each case, see Townsend and Altieri (2012). The `assessment` function with the **sft** package can be used for detection tasks with the following syntax:

```
assessment(RT, CR, OR, correct, fast, detection=TRUE)
```

The `RT` and `CR` are lists of response times and correct indicators for each trial. As in the standard capacity R functions, the first element in the list contains the measurements from trials in which both sources of information were present and the second and third elements are for each of the single-source conditions. The `OR` input is a `TRUE/FALSE` indicator of whether to calculate the assessment function using an UCIP-OR baseline (`OR=TRUE`) or an UCIP-AND baseline (`OR=FALSE`). The correct and fast parameters are `TRUE/FALSE` indicators to specify which of the four types of assessment functions to use.

For example, to evaluate a participant (S7) from the OR-decision dot detection task, we first extract the necessary data,

```
sub <- 'S7'
cond <- 'OR'
#select single channel data
chan1 <- dots[Subject==sub & Condition==cond & Channel1>0 & Channel2==0,
c('RT', 'Correct')]
chan2 <- dots[Subject==sub & Condition==cond & Channel1==0 & Channel2>0,
c('RT', 'Correct')]
#select redundant target (2-channel) data
redundant <- dots[Subject==sub & Condition==cond & Channel1>0 & Channel2>0,
c('RT','Correct')]
rts <- list(redundant$RT,chan1$RT, chan2$RT)
corrects <- list(redundant$Correct, chan1$Correct, chan2$Correct)
```

Next, we simply apply the function:

```
a.or.cf <- assessment(rts, corrects, OR=TRUE, correct=TRUE, fast=TRUE,
detection=TRUE)
```

The output is a stepfun object, so it can be plotted using plot:

```
plot(a.or.cf, ylim=c(0,2))
```

Figure 4 shows each of the correct/incorrect and fast/slow assessment functions for Participant 7 in the OR condition. Note that UCIP performance would show a value of 1 for all times in all plots.

In discrimination OR tasks, a participant may respond based on whichever source finishes first. Hence, the response will be incorrect if the first to finish is incorrect even if the second source would have been correct. This results in a slightly different baseline for performance assessment. Now, for a correct response, either T_{AC} or T_{BC} must be faster than both T_{AI} and T_{BI}. The UCIP baseline for correct-fast, OR, discrimination is:

$$\int_0^t f_{AC}(t)\,[1-F_{AI}]\,[1-F_{BI}] + \int_0^t f_{BC}(t)\,[1-F_{AI}]\,[1-F_{BI}]$$
$$-\int_0^t f_{AC}(t)\,[1-F_{AI}]\,[1-F_{BI}]\int_0^t f_{BC}(t)\,[1-F_{AI}]\,[1-F_{BI}]$$

See Donkin, Little, and Houpt (2013), particularly the appendix, for details of the discrimination assessment functions. The R syntax for discrimination tasks is the same as the syntax for the detection task, but with the `detection` parameter set to `FALSE`.

Conclusion

Workload capacity analysis entails a powerful set of tools within SFT for examining the effects on information processing of differing numbers of information sources (different numbers of stimulus inputs, different numbers of active processing channels). Several recent theoretical additions to capacity analyses have both expanded the applicability of

capacity to a new stopping rule (ST-ST processing) and broadened the available tools for capacity analysis, especially to allow more nuanced comparisons across participants and experimental conditions. Despite being a powerful framework based on minimal assumptions (and often relying on non-parametric analyses), SFT is underutilized within the psychological research community, partly because researchers previously needed to develop their own computational codes. We hope that by making the tools accessible with open source R functions and with the present paper together with Houpt et al. (2013), researchers can easily use the SFT tools more frequently.

Here, we have described briefly the new theoretical advances and provided a detailed account of the new functions for utilizing the new tools in the R statistical computing framework. These new functions constitute the first major additions to the **sft** package beyond the initial functionality described in Houpt et al. (2013). The advantage of this paper is that it focuses on the computational implementation for using the new capacity tools with detailed examples of the R code. Researchers seeking to try capacity analysis now have a standardized implementation of these functions, together with the other SFT tools for assessing processing architecture made available in the **sft** package. We encourage researchers to use this standardized R package to reduce the chance of implementation errors that inevitably arise when each user is left to themselves to translate from a theoretical paper to usable code. And as additional theoretical advances are made in SFT, we will continue to update the **sft** package as the state of the science for SFT modeling.

References

Blaha, L. M. (2010). *A dynamic Hebbian-style model of configural learning.* Unpublished doctoral dissertation, Indiana University, Bloomington, Indiana.

Blaha, L. M., & Houpt, J. W. (Under Review). Generalized n-channel workload capacity space.

Blaha, L. M., & Townsend, J. T. (under review). On the capacity of single-target, self-terminating processes.
(Manuscript submitted for publication)

Burns, D. M., Houpt, J. W., Townsend, J. T., & Endres, M. J. (2013). Functional principal components analysis of workload capacity functions. *Behavior Research Methods*, *45*, 1048-1057. doi: 10.3758/s13428-013-0333-2

Colonius, H., & Vorberg, D. (1994). Distribution inequalities for parallel models with unlimited capacity. *Journal of Mathematical Psychology*, *38*, 35–58.

Donkin, C., Little, D., & Houpt, J. W. (2013). Assessing the speed-accuracy trade-off effect on the capacity of information processing. *Journal of Experimental Psychology: Human Perception and Performance.*

Eidels, A., Donkin, C., Brown, S. D., & Heathcote, A. (2010). Converging measures of workload capacity. *Psychonomic Bulletin & Review*, *17*, 763-771.

Houpt, J. W., Blaha, L. M., McIntire, J. P., Havig, P. R., & Townsend, J. T. (2013). Systems Factorial Technology with R. *Behavior Research Methods.* (Advance online publication) doi: 10.3758/s13248-013-0377-3

Houpt, J. W., & Townsend, J. T. (2012). Statistical measures for workload capacity analysis. *Journal of Mathematical Psychology*, *56*, 341-355.

Miller, J. (1982). Divided attention: Evidence for coactivation with redundant signals. *Cognitive Psychology*, *14*, 247-279.

R Development Core Team. (2011). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from

`http://www.R-project.org` (ISBN 3-900051-07-0)

Raab, D. (1962). Statistical facilitation of simple reaction times. *Transactions of the New York Academy of Sciences, 24*, 574-590.

Ramsay, J. O., Hooker, G., & Graves, S. (2009). *Functional data analysis with R and MATLAB.* New York: Springer.

Ramsay, J. O., & Silverman, B. W. (2005). *Functional data analysis* (2nd ed.). New York: Springer.

Townsend, J. T. (1974). Issues and models concerning the processing of a finite number of inputs. In B. H. Kantowitz (Ed.), *Human information processing: Tutorials in performance and cognition* (p. 133-168). Hillsdale, NJ: Erlbaum Press.

Townsend, J. T., & Altieri, N. (2012). An accuracy-response time capacity assessment function that measures performance against standard parallel predictions. *Psychological Review, 119*, 500-516.

Townsend, J. T., & Ashby, F. G. (1983). *The stochastic modeling of elementary psychological processes.* Cambridge: Cambridge University Press.

Townsend, J. T., & Eidels, A. (2011). Workload capacity spaces: A unified methodology for response time measures of efficiency as workload is varied. *Psychonomic Bulletin & Review, 18*, 659-681.

Townsend, J. T., & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Townsend, J. T., & Wenger, M. J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003-1035.

Wenger, M. J., & Townsend, J. T. (2000). Basic tools for attention and general processing capacity in perception and cognition. *Journal of General Psychology: Visual Attention, 127*, 67-99.

Table 1

Summary of all Bounds on the Capacity Coefficient (from Blaha & Houpt (under review))

LOWER BOUNDS

Stopping Rule	n-channels	n IID channels	2 channels
OR	$\frac{\ln\{\min_i\left[S_{C\setminus\{i\}}(t)\right]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{S_{C\setminus\{1\}}(t)\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{\min[S_1(t),S_2(t)]\}}{\ln\{S_1(t)*S_2(t)\}}$
STST	$\frac{\ln\{F_k(t)\}}{\sum_{c=1}^{n}\ln\{F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{n*\ln\{F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{F_1(t)*F_2(t)\}}$
AND	$\frac{\ln\left(\prod_{c=1}^{n} G_c(t)\right)}{\ln\{\max_{i,j}\left[G_{C\setminus\{i\}}(t)+G_{C\setminus\{j\}}(t)-G_{C\setminus\{i,j\}}(t)\right]\}}$	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{2*G_{C\setminus\{1\}}(t)-G_{C\setminus\{1,2\}}(t)\}}$	$\frac{\ln\{G_1(t)*G_2(t)\}}{\ln\{G_1(t)+G_2(t)-1\}}$

UPPER BOUNDS

Stopping Rule	n-channels	n IID channels	2 channels
OR	$\frac{\ln\{\max_{i,j}\left[S_{C\setminus\{i\}}(t)+S_{C\setminus\{j\}}(t)-S_{C\setminus\{i,j\}}(t)\right]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{2*S_{C\setminus\{1\}}(t)-S_{C\setminus\{1,2\}}(t)\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{S_1(t)+S_2(t)-1\}}{\ln\{S_1(t)*S_2(t)\}}$
STST	$\frac{\ln\{F_k(t)\}}{\ln\{\sum_{c=1}^{n} F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{n*F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{F_1(t)+F_2(t)\}}$
AND	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\min_i\left[G_{C\setminus\{i\}}(t)\right]\}}$	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{G_{C\setminus\{1\}}t)\}}$	$\frac{\ln\{G_1(t)*G_2(t)\}}{\ln\{\min[G_1(t),G_2(t)]\}}$

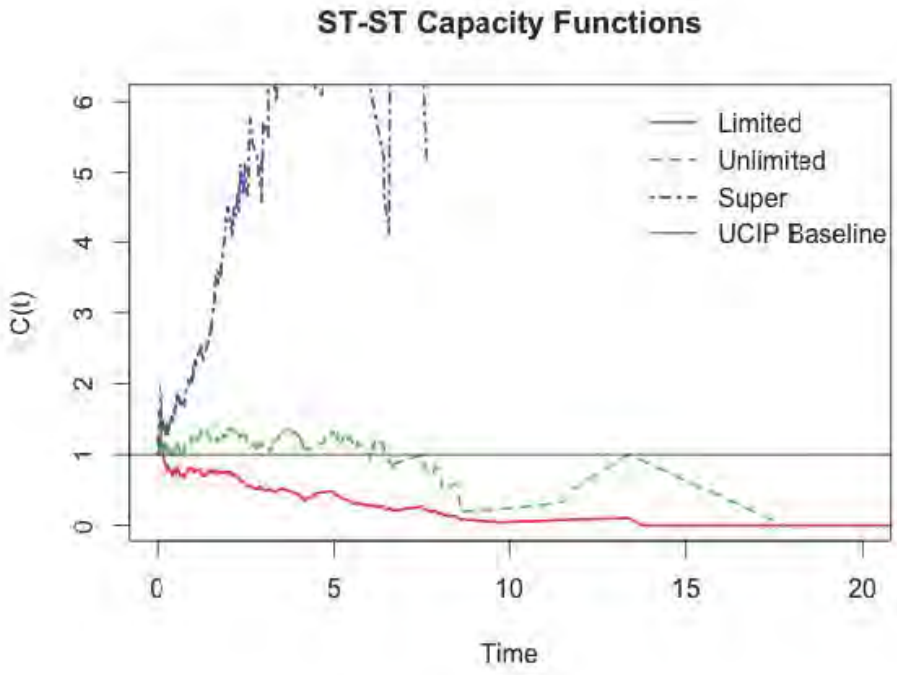

Figure 1. Plots of ST-ST processing capacity coefficients, in ratio=TRUE form. The data were simulated from ST-ST processing, including a model exhibiting limited, unlimited, and super capacity processing rates, and the corresponding C_{STST} estimates are plotted in red, green, and blue (respectively). The baseline reference model, giving $C_{STST} = 1$ is plotted in the thin, black line.

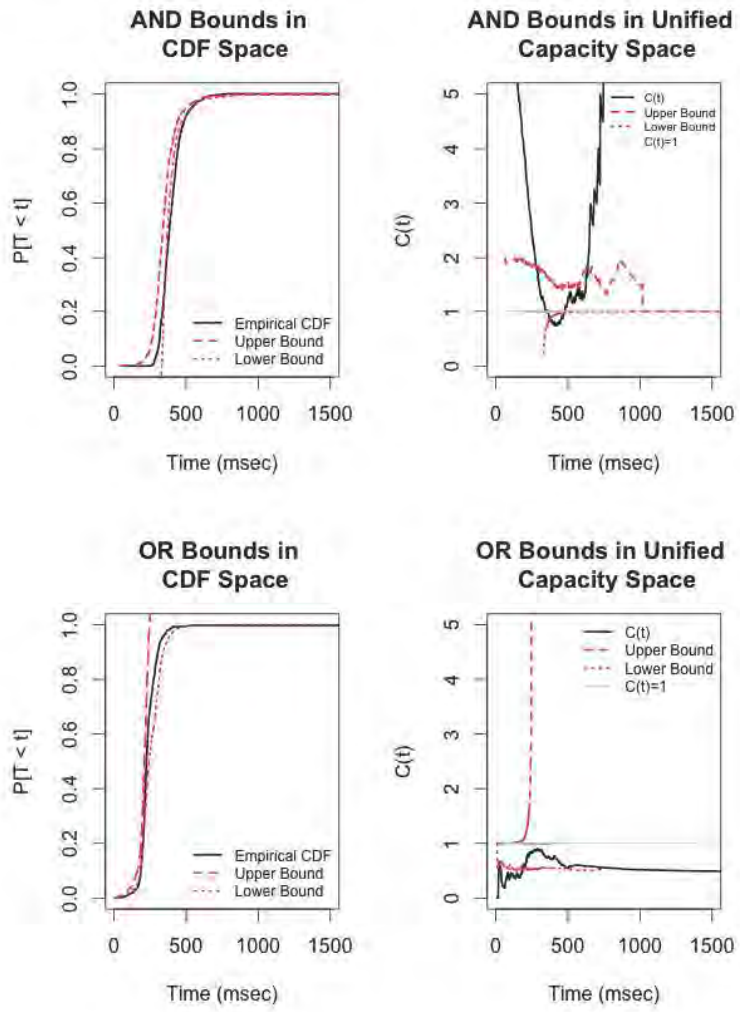

Figure 2. Example bounds on standard parallel processing from one participant (S3) in the dots data included in the **sft** package. The top row shows the bounds for AND processing, and the lower row illustrates the bounds for OR processing. The left hand plots give the traditional CDF space plots, with the bounds on the CDF for the redundant signals response times. The right hand plots show the newer unified capacity space version of the same bounds, plotted against the empirical capacity coefficient function.

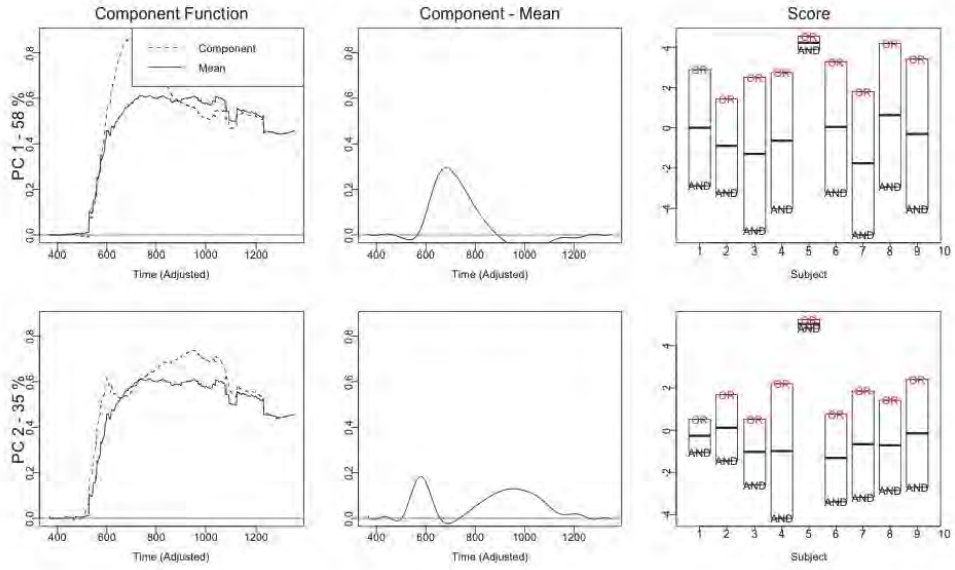

Figure 3. Sample fPCA plots computed on the dots data included in the **sft** package. The far left plots show the component functions together with the mean capacity function; the center plots show the difference between the component and the mean capacity functions. The right-hand plots show the loading scores for each participant (x-axis) and for each experimental condition (here, termed OR and AND).

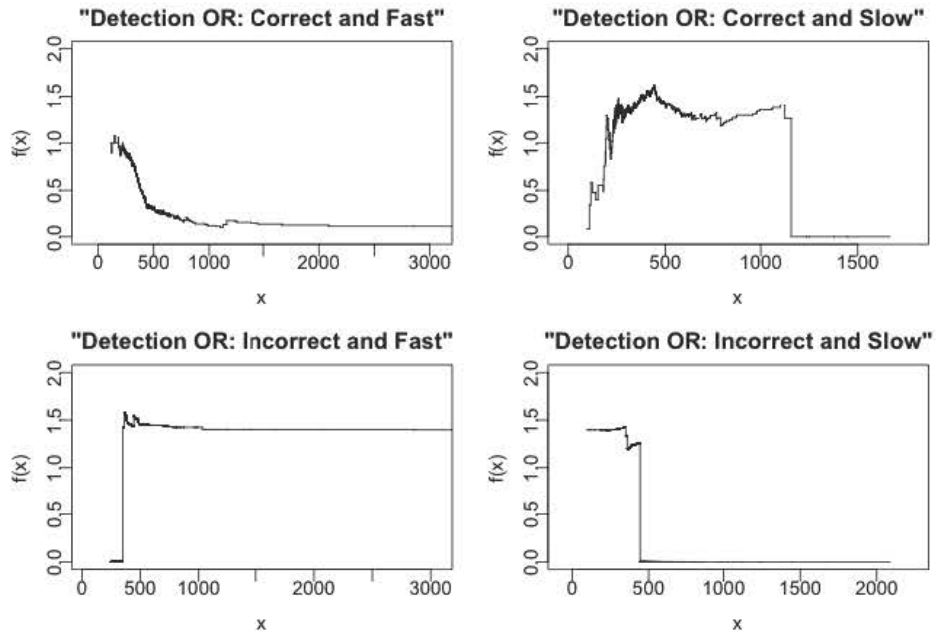

Figure 4. Sample assessment function plots computed on one participant (S7) in the the dots data included in the **sft** package.

Noname manuscript No.
(will be inserted by the editor)

Opinions, Influence, and Zealotry

Author Name Omitted

the date of receipt and acceptance should be inserted later

Abstract A novel dimer automaton model for innovation diffusion based on a simplification of the AB model and zealot model is proposed. The model assumes that two opposing opinions are competing to be the dominant opinion among individuals in a network. Zealots are stubborn individuals whose opinion is not susceptible to influence by others. The amount of zealots required for consensus is measured experimentally in a number of different situations. The threshold density of zealots is far lower than the control experiment, suggesting that zealots have a much larger influence than normal individuals in the model. This threshold can be further reduced by placing zealots at critical nodes in the network, determined by standard social network measures or by using a greedy algorithm. Other experiments show that when both opinions have zealots, the outcome depends on the total number of zealots in addition to the ratio of zealots of opposite opinion, and often results in an "undecided" outcome.

Address(es) of author(s) should be given

1 Introduction

Innovation diffusion addresses the adoption of new technologies throughout society [20]. Since its introduction, the concept has been applied to a number of different domains not originally envisioned, and innovation diffusion is often used as a metaphor to describe any number of things (technologies, opinions, attitudes, decisions) that spread through a population. The innovation rate (i.e., the number of individuals who adopted the new technology) over time typically follows logistic-like growth (i.e., growing exponentially, and then slowing as the innovation nears full adoption). Ideally, from a marketing standpoint, understanding innovation diffusion helps answer the question "how do I ensure my product takes off?" Many studies have looked at this problem in hindsight, but general purpose, accurate, and reliable predictors are not currently available.

This paper introduces a new individual modeling and simulation approach for innovation diffusion that is predictive for a certain class of idealized, but realistic scenarios. The proposed model, which is certainly a gross oversimplification of human behavior, allows an individual to have a state taken from a small finite set of possible states. Individuals change their states over time by interacting with other individuals in a pairwise fashion according to a deterministic rule (however the order of interactions is random). Interactions are assumed to occur only between adjacent individuals in the user-defined network. Despite the limitations of this oversimplified individual model, there are several advantages worth highlighting.

First, and from a practical standpoint, the simplicity allows for a very efficient computer implementation. For example, a million simulations, each with ten thousand individuals, were completed in a few minutes using the proposed model on

a single workstation with an adequate CPU. Second, models designed to be "realistic" often become so complex that it is difficult or impossible to reason about which components are most directly influencing the observed behavior, or are even important to the model. Simple models are more easily communicated between researchers in disparate areas, and can be implemented and modified with little effort to produce new results. Furthermore, the fine details of individual complexity tend to "wash out" when one considers the collective behavior of populations. Use of a simpler model can help circumvent these issues. Finally, a simpler model is more amenable to future rigorous analytical treatment, especially if it can be shown that the model elegantly captures some interesting behavior. Thus, these advantages make simple individual models attractive for use in large scale simulations, which are necessary to understand and predict the collective behavior of individuals.

1.1 Related Work

Threshold models were one of the earliest attempts to understand how individual variations throughout a population affected the innovation diffusion curve [12]. These models assume that each individual has complete information about all other individuals and has some threshold for taking action based on this information. However, the assumption for individuals to have complete information may not always be appropriate, so relaxation of this assumption led to models such as the Linear Threshold Model [24]. In this model, a individual has a state encoding whether they have or have not adopted. Once adopted, the individual cannot un-adopt, so the diffusion is progressive. With the Linear Threshold Model, individuals adopt if the fraction of neighbors having adopted is larger than their

given threshold. The threshold can be randomly assigned, or fixed (e.g., to 1/2). Models like this strive to represent the behavior of an individual in a way that allows the collective behavior of the population to be an emergent property of the system. The power of individual models is that, when successful, they illuminate the relationship between individual actions and collective outcomes.

There are many other individual models of social dynamics including broad areas such as opinions, cultures, languages, and crowds [3]. Another adoption model, The Independent Cascade Model, assumes a stochastic flavor, giving each newly adopting individual one opportunity to influence each of it's neighbors according to some probability [10]. The voter model is a simple and popular model for opinion dynamics [13]. In this model, one picks a vertex at random and the state of that vertex is then changed to take on the state of a randomly chosen neighbor, which performs coarsening via interface noise. There have been many variants and explorations into this simple model. The ideas presented in this paper are based on the zealot variant [21,22] and the centrist/AB variant [26,4]. In the zealot variant, some vertexes are "zealots" and have a bias towards one opinion over the other. The existence of a few zealots can significantly affect the long term outcome of the system. In the centrist/AB variants, an additional intermediate state is introduced, and it is assumed that states cannot change without first passing through the intermediate state (i.e., in order to change from A (left) to B (right) one must first become AB). In the AB model, the probability that $A \to AB, AB \to B$, etc., is based on the neighborhood density of A, B, and AB.

It is known that models with intermediate states like the AB model accomplish coarsening by reducing the surface tension along the boundary between opposing domains. Such models are sometimes referred to as "curvature driven" models,

as opposed to interface noise models. Furthermore, the models discussed above are different from rumor and epidemic models, since the opinions compete for territory versus quickly spreading within vulnerable regions as epidemic models do. The model presented in this paper is a combination of the zealot and the AB model, and a simplification of both.

Given a model of influence and opinion or adoption like those discussed above, is it possible to determine a small set of individuals that, when influenced, can catalyze change throughout the entire network? This question is at the heart of the research area of Influence Maximization [8]. A solution close to optimal is very valuable in a marketing context, for example, as it could lead to an effective allocation of advertising resources. The current basis for influence maximization techniques is to assume an adoption model like the Linear Threshold Model [24] or the Independent Cascade Model [10] and compute the smallest set of seeds that will cause adoption to spread throughout the entire network.

The greedy algorithm by Domingos et al., works by computing the spread of influence throughout the network for a given set (which is initially empty), and finding the individual (who is not in that set) that increases the spread of influence the most [8]. That individual is chosen and added to the set, and the algorithm repeats until the influence has covered the entire network, with the solution being the set after termination of the algorithm. Kempe et al. later proved that the greedy algorithm will reach within 63% of optimal for these models [15]. Because the greedy algorithm is effective, but computationally expensive, researchers have developed techniques that improve the efficiency of influence maximization techniques[27,6,11,19].

1.2 Proposed Model

The model to be proposed here is a dimer automaton model of opinion dynamics involving zealots and curvature-driven coarsening via an intermediate state. Dimer automata are similar to voter models; however, instead of updating one vertex at a time, one edge is chosen per asynchronous update step. For this reason a dimer automaton can be thought of as pattern matching and substitution system. Both endpoints of that edge may be simultaneously changed, avoiding the asymmetry problem with the voter model [3]. Formally, we assume some graph $G = (V, E)$ where V and E can be interpreted as the individuals and their relationships in the model, respectively. Let x_i^t be the state of vertex (individual) i at time t. To perform an update, an edge $(i, j) \in E$ is chosen at random, and the endpoints of the edge are updated symmetrically such that

$$x_i^{t+1} = R(x_i^t, x_j^t),\ x_j^{t+1} = R(x_j^t, x_i^t). \tag{1}$$

The application of the rule to x_i^t and x_j^t can be thought of abstractly as i and j interacting at time t. Also, t is simply a counter of the number of edges updated so far, and only one edge is updated at a time (but edges can be updated many times over through the course of the simulation). The extremely large space of rules for a given set of states is gives dimer automata the potential to model a wide range of phenomena. The rule behaves as a finite state automaton from the viewpoint of each x_i. For the opinion dynamics model for this paper, let the rule be defined as

$$R(\sigma, \tau) = \begin{cases} |\tau| \text{ if } \sigma = 0 \\ 0 \quad \text{if } \sigma > 0 \text{ and } \sigma \neq |\tau \text{ and } \tau \neq 0 \\ \sigma \quad \text{else} \end{cases}, \tag{2}$$

which is based off an earlier 3-state dimer automaton rule for domain coarsening [1]. This logic encoded in this rule is "generalized," meaning the rule can support an arbitrary number of possible opinions without any modification.

It is important to differentiate between the opinion of an individual and the state of that individual. An individual with state σ has opinion $|\sigma|$. Thus, the sign of the state designates whether that individual is a zealot or not (zealots are negative). State 0 acts as the intermediate (i.e. centrist/AB) state that positive states must pass through to change from one opinion to another. Since dimer automaton rules are deterministic, the proposed model is a simplification of the centrist/AB model. The allowable transitions are equivalent, but it is not necessary to know the how many neighbors have a particular state, which simplifies the model and improves the computational efficiency. Finally, it is worth noting that the meaning of "zealot" in a dimer automaton is slightly different than in the previous literature. Voter model zealots have a bias towards a particular opinion, which is implemented as an increased probability that the zealot will take on that state. However, a zealot in the dimer automaton model can be thought of having maximal bias towards a particular opinion (i.e., the probability the zealot takes on its favored opinion is 1). This is a result of dimer automaton rules being deterministic, as opposed to voter model rules which are probabilistic.

For clarity, consider the following example. Suppose there are two political parties referred to as "red" and "blue," which are equivalent to opinion 1 and opinion 2 respectively. Suppose Alice and Bob are friends (i.e., the edge $(Alice, Bob) \in E$ so the dimer automaton can randomly choose the edge connecting Alice and Bob and update their states). Let x_A^t and x_B^t refer to the state of Alice and Bob, respectively. If Alice and Bob are both red or both blue (i.e., $x_A^t = x_B^t$), then no

change occurs when they interact since $R(1,1) = 1$ and $R(2,2) = 2$. However, suppose Alice is red and Bob is blue (i.e., $x_A^t = 1$ and $x_B^t = 2$); after they interact, both Alice and Bob would become undecided and susceptible to influence (i.e., $x_A^{t+1} = x_B^{t+1} = 0$ since $R(1,2) = R(2,1) = 0$). It would then fall to another friend of Alice and/or Bob to reorient their affiliations. For example, suppose Eve is friends with Alice, and Eve is blue. Then, when Eve interacts with the undecided Alice, Eve persuades Alice to become blue (i.e., $x_A^{t+2} = 2$ since $R(0,2) = 2$). Thus, Alice has switched from red to blue through the influence of both Bob and Eve. This mechanism is what drives the curvature based dynamics since, on average, Alice will adopt the opinion of the majority of her neighbors.

The zealot is a simple mechanism intended to account for stubborn individuals, since a zealot never changes their opinion. In the political debate, it is generally accepted that a certain percentage of individuals will never change their political affiliation: in fact people may change their friends to suit their affiliations [5]. So, suppose this time that Alice is a red zealot, and Bob is still just blue (i.e., $x_A^t = \quad 1$ and $x_B^t = 2$). When Alice and Bob interact, Alice remains a red zealot, but Bob becomes undecided (i.e., $x_A^{t+1} = -1$ and $x_B^{t+1} = 0$ since $R(-1,2) = -1$ and $R(2,-1) \quad 0$). The same effect happens when Alice and Eve interact. If Alice and Bob interact again (with Bob now undecided), Bob will be recruited over to red from undecided, however Bob never becomes a zealot (i.e., $x_A^{t-2} = -1$ and $x_B^{t-2} = 1$ since $R(-1,0) = -1$ and $R(0,-1) = 1$). The rule is designed such that non-zealots never become zealots, and zealots never become non-zealots.

2 Experiments

2.1 Control

We are interested in understanding how opinions collectively change over time, and how this change depends on the initial configuration of the system (i.e., x_i^0 for each $i \in V$). For this section we consider a simple case where the system is almost entirely non-zealot blue, aside for a handful of red zealots. Each zealot is assigned to a randomly chosen vertex in the graph. Does the system reach a consensus[1] after a reasonable amount of time? An example[2] of this is shown in Fig. 1. The four snapshots show the configuration of the system after the application of Eqn. 2 millions of times. Initially the system consists only of blue states (shown as white) and a few red zealots (shown as black), but the zealots are able to quickly spread their influence and dominate the entire system.

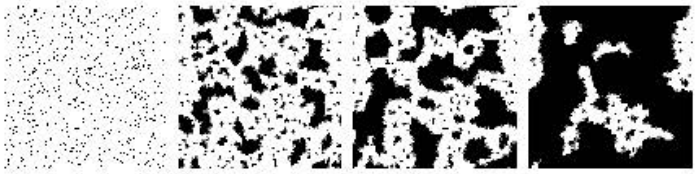

Fig. 1 The configuration of the system over time (moving from left to right) shows the consensus transitioning from opinion 1 (white) to opinion 2 (black).

[1] Consensus is measured as the ratio of the number of opinion 2 non-zealots to total non-zealots. Zealots are left out of this ratio since the population is known at the start of the simulation and does not change.

[2] The graph used is a 100×100 square lattice with von Neumann neighborhoods and periodic boundary conditions, since this has a straightforward visualization.

For subsequent experiments in this section we use a Watts-Strogatz small world network [29] with rewire probability 0.1 and size 100 × 100. Fig. 2 shows how the consensus changes over time. The dynamics are more complicated than classical population-based models of innovation diffusion, which often follow a logistic curve. The system goes through a period of slowing growth, then quickening growth, and again slowing as consensus is nearly reached. This curve exhibits two inflection points, as opposed to the logistic curve which has only one.

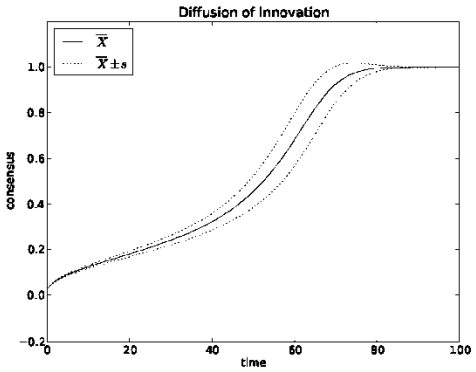

Fig. 2 The average consensus over time has two inflection points, a more complex and realistic behavior than the typical logistic curve associated with innovation diffusion.

2.2 A Simple Experiment with Zealots

What effect, if any, do zealots have on the system, and how do we measure this? To begin, we must first run a control experiment with no zealots present, and observe the outcomes of different ratios of initial opinions. In other words, what is the

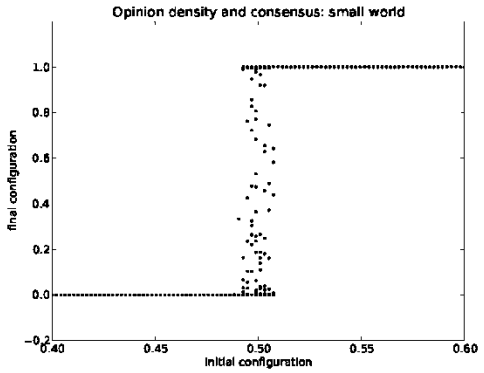

Fig. 3 Control experiment varying the initial density of opinion 1 and 2 (no zealots) for a small world network (Watts-Strogatz, $d = 2, p = 0.1$).

outcome starting with mostly red versus mostly blue? Fig. 3 shows the outcome of 9216 experiments[3] with varying opinion densities in the initial configuration. There is only a small window centered around 0.5 (i.e., equal quantities of opinion 1 and 2) where the density of the final configuration is between 0 or 1 (i.e., the outcome is uncertain). So, 0.5 appears to be a critical point for the system with any density slightly above or below moving quickly to 1 or 0. Based on this, we can let 0.5 be a reasonable threshold to determine whether or not the zealots have taken over the system. In other words, once an opinion is held by more than than half the population, that opinion tends to quickly take over the rest of the network.

Now we can determine what initial density of zealots is necessary to shift the consensus from the prevailing opinion to the opinion of the zealots. If zealots only exert short range influence, then the control suggests the threshold for consensus

[3] Experiments were efficiently conducted in parallel on the GPU using the technique described in [2].

would remain close to 0.5. Repeating and averaging a number of independent trials for a range of zealot densities tests this hypothesis. For each experiment and once the number of zealots are determined, each zealot is assigned to random vertex in the network. We define the "critical zealot density" $Z_\star$ as the initial zealot density that produces a consensus above ϵ, and for this experiment let $\epsilon = 0.5$. This quantity is computed in a straightforward manner according to Algorithm 1. An example of this measurement is shown in in Fig. 4, where the order provided to the algorithm was a random permutation of the nodes in the network. Surprisingly, we can see that $Z_\star$ (approximately 0.074, shown by the dotted line) is nearly an order of magnitude lower than the density observed in the control. Zealots have a much higher influence on the outcome than expected.

Algorithm 1 Compute $Z_\star$ for a given order.

Let $(v_1, v_2, ..., v_{|V|})$ be an ordering of the vertices in the network
min i
s. t. CONSENSUS$(i) > \epsilon$
$Z_\star = i_\star/|V|$
procedure CONSENSUS(i)
    $X := (1, 1, ..., 1)$ ▷ has length $|V|$
    $X(v_1, v_2, ..., v_i) = -2$ ▷ assign zealots based on order
    run an experiment with X_0 as the initial configuration
    measure the consensus at the end of the experiment
    return $\frac{1}{|V|}\sum_i \delta(2 - |x_i|)$ ▷ measure consensus
end procedure

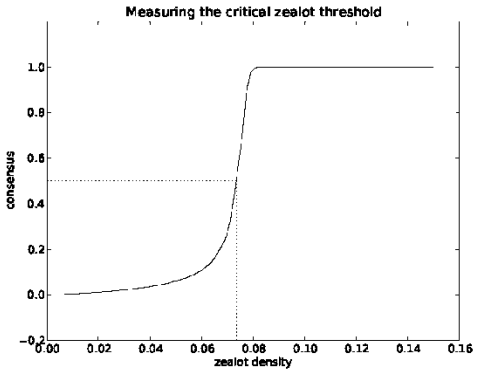

Fig. 4 Determining the critical zealot threshold Z_* by measuring when consensus passes 0.5.

2.3 Varying Network Structure

The previous experiment is repeated with different graphs to determine the effect of the network used on the critical zealot density. The Watts-Strogatz small world network [29] is a common way to explore how a model or phenomena is affected by network structure. This model defines a rewire probability p, which generates networks that transition between uniformity (e.g., a square lattice) and randomness. From Fig. 5 we can see that the graph has an interesting effect on the critical zealot threshold. As the rewire probability is increased (and the network becomes more disorganized) Z_* increases quickly. However, this threshold appears to level out and does not surpass 0.1, even for a fully disorganized network. From this we can conclude that the network structure has a significant effect on Z_*, so subsequent experiments consider a variety of networks.

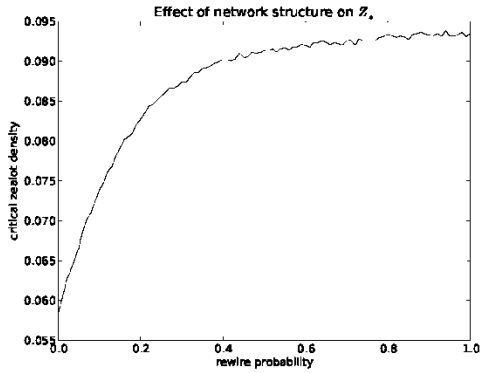

Fig. 5 As the network transitions from order to disorder, the critical zealot density increases.

2.4 Influence Maximization

Influence maximization is a useful application for models of opinion dynamics such as the one proposed in this paper. Given a model and network, influence maximization helps us find a small set of individuals that can precipitate a change throughout the entire network [8]. For the zealot dimer automaton model proposed in this paper, the problem of influence maximization translates into finding the optimal set of nodes in the network that should start as zealots in the initial configuration. Past research in influence maximization has shown that the greedy algorithm outperforms random selection as well as other heuristics based on social network analysis measures such as closeness, betweenness, and degree. The purpose of the following experiment is to determine whether this result also holds for the zealot dimer automaton model.

Based on the experiment in the previous section, we know that the structure of the network can have a significant effect on the consensus threshold Z_*, even if

Table 1 Networks for Influence Maximization Experiment

name	$\|V\|$	$\|E\|$	details	ref(s)
wiki-Vote	7115	103689	who votes on whom for Wikipedia adminship elections	[17,16]
ca HepTh	9877	25998	High Energy Physics–Theory arXiv collaboration network	[18]
ca-GrQc	5242	14496	General Relativity and Quantum Cosmology arXiv collaboration network	[18]
Power Law Cluster	10000	29990	random scale free network with $m = 3, p - 0.1$	[14]
Erdős-Rényi	10000	39608	random graph with $p = 1.23 \times 10^{-3}$	[9]
Watts-Strogatz	10000	20000	random small world network with $k = 9, p = 0.2$	[29]

zealots are chosen randomly. Therefore, the following experiment considers several types of networks (see Table 1) as well as several different heuristics for influence maximization. Heuristics are based on centrality metrics from social network analysis centrality: degree, closeness, and betweenness [28]. Degree centrality simply measures the number of neighbors adjacent to a given node. Closeness centrality is the inverse of the average distance for a given node to all other nodes. Betweenness centrality considers the fraction of all shortest paths that pass through a given node. Each of these metrics are measured for all nodes in the network to determine a ranking. These metrics determine an ordering of the nodes in the network, which are used by Algorithm 1 to compute the critical zealot density resulting from that particular ordering.

These heuristics are compared against a variation of the classical greedy algorithm for influence maximization [8], adapted for use with the zealot dimer automa-

Table 2 Summary of Results for Influence Maximization Experiment

name	betweenness	random	closeness	degree	greedy
Erdos Renyi	6.88×10^{-2}	1.13×10^{-1}	7.54×10^{-2}	$\mathbf{6.43 \times 10^{-2}}$	8.44×10^{-2}
Watts Strogatz	6.00×10^{-2}	6.23×10^{-2}	9.46×10^{-2}	4.02×10^{-2}	$\mathbf{3.50 \times 10^{-2}}$
Power Law Cluster	9.10×10^{-3}	1.14×10^{-1}	1.20×10^{-2}	$\mathbf{9.00 \times 10^{-3}}$	9.60×10^{-3}
ca-GrQc	1.81×10^{-2}	9.82×10^{-2}	5.30×10^{-2}	3.78×10^{-2}	$\mathbf{1.37 \times 10^{-2}}$
ca-HepTh	1.04×10^{-2}	8.22×10^{-2}	1.73×10^{-2}	$\mathbf{8.00 \times 10^{-3}}$	1.49×10^{-2}
wiki-Vote	9.70×10^{-3}	1.21×10^{-1}	8.57×10^{-3}	$\mathbf{8.15 \times 10^{-3}}$	1.14×10^{-2}

ton model, which is outlined in Algorithm 2. This algorithm starts with an initial configuration X_0 and a set of allowable moves encoded in $M = \{(v_1, \sigma_1), (v_2, \sigma_2)...\}$, where the k^{th} move changes the state of node v_k to σ_k in the initial configuration. In the simplest case where we start with all opinion 1 and want to see how many opinion 2 zealots are needed to reach the critical threshold, we would let $X_0 = (1, 1, ..., 1)$ and $M = \{(i, -2) \ : \ i \in V\}$. Algorithm 2 also requires an objective function ϕ to minimize. In this case ϕ measures the consensus by counting the nodes in the network not having opinion 2, thus

$$\phi = \frac{1}{|V|} \sum_{i \in V} 1 \quad \delta(2 \quad |x_i|). \tag{3}$$

The results of the comparison for each graph and heuristic are shown in Figure 6, and a summary of the critical zealot densities is provided in Table 2. Surprisingly, no single heuristic nor the greedy algorithm is a clear winner, though, the random heuristic usually results the worst critical density. Another interesting feature seen in Figure 6 is that the greedy algorithm tends to dominate the other heuristics early in the simulation, but may not be the first to reach criticality.

Algorithm 2 Greedy algorithm for influence maximization for the zealot model

X_0 is the initial configuration of the network
ϕ is an objective function that scores the configuration of the network
M is the list allowable moves
$M :=$ current random number generator state
while convergence criteria not met **do** ▷ can be an arbitrary threshold ϵ
    $k_+ := \underset{k}{\operatorname{argmin}}\ \textsc{score}(X_\circlearrowright, k)$
    $(i, \sigma) = M(k)$
    $X_\circlearrowright(i) = \sigma$
end while
procedure $\textsc{score}(X, k)$
    set random number generator state to M
    $(i, o) = M(k)$
    $X(i) := o$
    $X_f :=$ result of experiment with initial input of X
    return $\phi(X_f)$
end procedure

2.5 Competitive Zealotry and Political Polarization

Realistically, we can expect to encounter both individuals who advocate for change, and those who resist it. In other words, we should investigate scenarios where zealots are present for both opinions. Clearly the outcome will depend on the ratio of these two types of zealots, but it may also depend on the total quantity of both types of zealots as well. Fig. 7) measures the consensus of the system for 96^2 different pairs of zealot densities in the range from 0 to 0.3. For each pair, the experiment is rerun 100 times for a reasonable sample size. In this figure, a line is drawn showing when the consensus crosses 33% and 66%, which separates the diagram into three distinct regions.

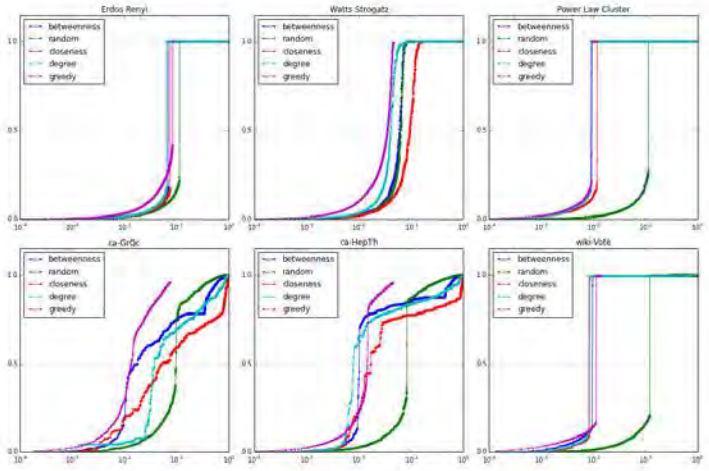

Fig. 6 The greedy algorithm for influence maximization is compared against rankings based on social network analysis metrics for several different graphs.

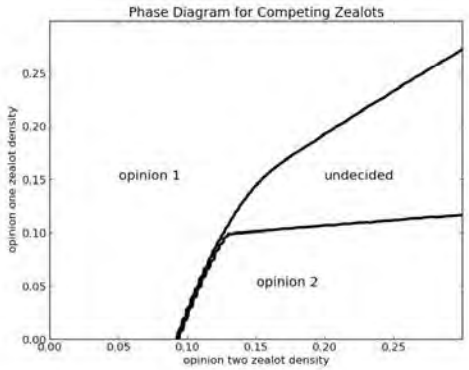

Fig. 7 Phase diagram for multi zealot experiment on Barabási-Albert network

If the consensus was solely dependent on the ratio of competing zealots, we would expect the transition between opinion 1 and 2 to follow a straight line whose slope was equal to the critical ratio. However, this transition follows a curved line. Furthermore, when the quantity of both zealots passes a certain threshold, the outcome becomes meta-stable (i.e., a stable combination of both opinions), which we refer to this as the "undecided" phase). When the density of opinion 1 zealots is low (e.g., < 0.1), the system chooses only between red and blue. However, with high enough densities of red and blue zealots, the system tends to remain in an undecided state, with significant amounts of both opinions present. This suggests that this model may be applicable to phenomena like political polarization where opposing opinions are held by significant fractions of the population.

To test this, we apply the influence maximization algorithm for the zealot model to a dataset consisting of politically charged communications between users of social media [7]. In addition to containing a social network, each node in the dataset is labeled as either left or right leaning, providing ground truth about opinions that can be leveraged. Applying influence maximization to this dataset requires some modifications since we are now maximizing influence for more than one target opinion. First, we assume that the initial configuration consists entire of some arbitrary third opinion, thus $X_0 = (3, 3, .., 3)$. Now, assuming that X_T is the target configuration of opinions in the network (i.e., the ground truth), the objective function becomes

$$\phi = \sum_{i \in V} 1 - \delta(|x_i| - X_T(i)) \tag{4}$$

and the set of allowable moves are $M = \{(i, -X_T(i)) \ : \ i \in V\}$.

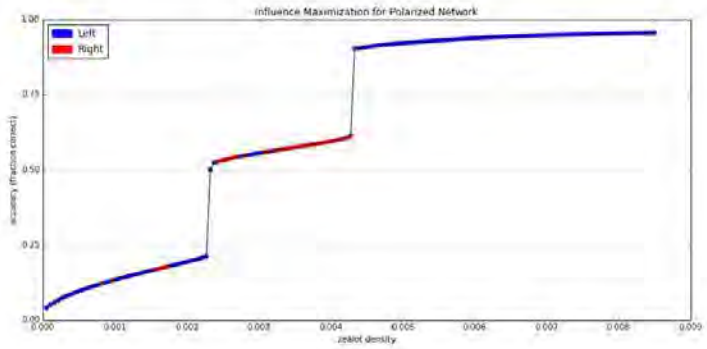

Fig. 8 default

3 Discussion

4 Conclusions & Future Work

The dimer automaton model presented combines and simplifies two variants models of opinion dynamics: the zealot model and the AB model. The resulting coarsening phenomena is curvature driven, and is used to investigate innovation diffusion. Using this model, we investigated some basic questions, namely, how many zealots are needed to reach consensus? The critical threshold of zealots required was significantly lower than 0.5, meaning only a few zealots in random locations in the network can significantly influence the entire system. This threshold depends on the network structure and the initial placement of the zealots in the network.

We also considered the case where both opinions have zealots, and some combination of zealots of both opinions leaves the system in an undecided state. Thus, the presence of individuals who refuse to change (e.g., opinion 1 zealots) could be an explanation of why some innovations fail to take hold.

A further challenge is to then verify these results against actual data such as marketing trials or elections. Real world data is often incomplete or contains uncertainty, so, an additional path for future work is to incorporate this into the model, perhaps by biasing how edges are randomly chosen by the dimer automaton according to a given probability distribution. Additionally it may be reasonable to upgrade the model so an individual's state lies on some spectrum between the two extremes instead of being a sharp choice between two opposing opinions. Hopefully this can be done in a manner that preserves the simplicity and elegance of the original model. This approach may be necessary if the simple model presented in this paper is not sufficiently predictive for real world data and scenarios.

References

1. Arendt, D., Cao, Y.: Evolutionary motifs for the automated discovery of self organizing dimer automata. Advances in Complex Systems **15**(07) (2012)
2. Arendt, D., Cao, Y.: GPU acceleration of many independent mid sized simulations on graphs. In: 4th Cellular Automata, Theory and Applications Workshop (* A CSC'12) (2012)
3. Castellano, C., Fortunato, S., Loreto, V.: Statistical physics of social dynamics. Reviews of Modern Physics **81**(2), 591–646 (2009)
4. Castelló, X., Eguíluz, V., San Miguel, M.: Ordering dynamics with two non-excluding options: bilingualism in language competition. New Journal of Physics **8**(12), 308 (2006)
5. Centola, D., Gonzalez-Avella, J.C., Eguiluz, V.M., San Miguel, M.: Homophily, cultural drift and the co-evolution of cultural groups. Journal of Conflict Resolution **51**(6), 905–929 (2007)
6. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery adn data mining, pp. 199–208. ACM (2009)

7. Conover, M., Ratkiewicz, J., Francisco, M., Gonçalves, B., Menczer, F., Flammini, A.: Political polarization on twitter. In: ICWSM (2011)
8. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 57–66. ACM (2001)
9. Erdős, P., Rényi, A.: On random graphs. Publicationes Mathematicae Debrecen **6**, 290–297 (1959)
10. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: A comlex systems look at the underlying process of word-of-mouth. Marketing letters **12**(3), 211–223 (2001)
11. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery adn data mining, pp. 47–48. ACM (2011)
12. Granovetter, M.: Threshold models of collective behavior. American Journal of Sociology **83**(6), 1420–1443 (1978)
13. Holley, R., Liggett, T.: Ergodic theorems for weakly interacting infinite systems and the voter model. The Annals of Probability **3**(4), 643–663 (1975)
14. HOlme, P., Kim, B.J.: Growing scale free networks with tunable clustering. Physical Review E **65**(2), 026,107 (2002)
15. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146. ACM (2003)
16. Leskovec, J., Huttenlocher, D., KLeinberg, J.: Predicting positive and negative links in online social networks. In: Proceedings of the 19th international conference on World wide web, pp. 641–650. ACM (2010)
17. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Signed networks in social media. In: Proceedings of the SIGCHI Conference on Human Facotrs in Computing Systems, pp. 1361–1370. ACM (2010)
18. Leskovec, J., Kleinberg, J., Faloustsos, C.: Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discoverey from Data (ACM TKDD) **1**(1), 2 (2007)

19. Leskovec, J., Krause, A., Guestrin, C., Faloutos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 420–429. ACM (2007)
20. Mahajan, V., Peterson, R.: Models for innovation diffusion, vol. 48. Sage Publications, Incorporated (1985)
21. Mobilia, M.: Does a single zealot affect an infinite group of voters? Physical Review Letters **91**(2), 028,701 (2003)
22. Mobilia, M., Georgiev, I.: Voting and catalytic processes with inhomogeneities. Physical Review E **71**(4), 046,102 (2005)
23. Newman, M.E.J.: Mixing patterns in networks. Physical Review E **67**(2), 026,126 (2003)
24. Schelling, T.C.: Micromotives and macrobehavior. WW Norton & Company (2006)
25. Solé, R.V., Valverde, S.: Information theory of complex networks: On evolution and architectural constraints. In: Complex Networks, pp. 189–207. Springer (2004)
26. Vazquez, F., Krapivsky, P., Redner, S.: Constrained opinion dynamics: Freezing and slow evolution. Journal of Physics A: Mathematical and General **36**(3), L61–L68 (2003)
27. Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery adn data mining, pp. 1039–1048. ACM (2010)
28. Wasserman, S., Faust, K.: Social network analysis: Methods and applications. Cambridge University Press (1994)
29. Watts, D., Strogatz, S.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)

Running head: GENERALIZED CAPACITY SPACE

Generalized n-Channel Workload Capacity Space

Leslie M. Blaha

Air Force Research Laboratory

Wright-Patterson AFB, Ohio

Joseph W. Houpt

Department of Psychology

Wright State University, Dayton, Ohio

Draft: January 30, 2014

Keywords: workload, capacity, human information processing, race model

Abstract

We provide the n-channel extension of the unified workload capacity space bounds for standard parallel processing models with minimum-time, maximum-time, and single-target self-terminating stopping rules. This extension enables powerful generalizations of this approach to multiple stopping rules and any number of channels of interest. Mapping the bounds onto the unified capacity space enables a single plot to be used to compare the capacity coefficient values to the upper and lower bounds on standard parallel processing in order to make direct inferences about extreme workload capacity.

Generalized n-Channel Workload Capacity Space

The study of the combination of multiple sources of information is ubiquitous in cognitive psychology. Examples include visual and memory search tasks, in which the multiple sources are the array items through which a participant must search, and complex decision making tasks in which multiple types of information must be combined to make a good decision. One question that often arises is the extent to which adding more sources of information affects the processing of each individual source. For example, one might inquire whether it takes longer to determine the presence of a particular object in a stimulus when there are more total objects in the stimulus. In this paper, we refer to a cognitive system's response to variations in the number of information sources as its workload capacity.

One of the most commonly used measures of workload capacity is the Race Model Inequality (Miller, 1982), which gives an upper bound on the response speed of a parallel processing model with context invariance (defined below) for testing one versus two sources of information using cumulative distribution functions (CDFs) in the context of minimum-time, redundant target decisions. Subsequent to Miller's paper, the basic logic of the Race Model Inequality has been extended to a develop lower bound on minimum-time models as well as upper and lower CDF bounds for other stopping criteria (e.g., all information must be processed rather than any one source) and more sources of information (Grice et al., 1984; Colonius & Vorberg, 1994). Using a stronger set of assumptions, together with a well-defined baseline model, Townsend and colleagues derived an equality to test workload capacity, termed the capacity coefficient (Townsend & Nozawa, 1995; Townsend & Wenger, 2004; Blaha & Townsend, under review).

Recently, Townsend & Eidels (2011) introduced the notion of a unified workload capacity space for plotting both the capacity coefficient and the CDF bounds on standard parallel processing on the same plot space. This work served to transform the upper and

lower bounds on parallel processing from probability space (ordinate values bounded on $[0, 1]$) into the same unit-less axis as the capacity coefficient, with ordinate values bounded on $[0, +\infty)$. Practically, this unified space allows investigators to directly compare *in the same plot* capacity coefficient values with the bounds on standard parallel processing, which enables some estimation of possible extreme capacity values (very high super capacity, very low limited capacity), as well as some inferences about possible model architectures (e.g. violation of the race model with super capacity implies a possible coactive model architecture). Unfortunately, Townsend & Eidels (2011) limited their derivations to models with only two possible operating channels. The capacity coefficients are defined for $n \geq 2$ channels (Townsend & Wenger, 2004), as are the CDF bounds on standard parallel processing (Colonius & Vorberg, 1994), so the restriction to $n \quad 2$ channels is an unnecessary limitation of the applicability for the new unified space.

Herein, we complete the derivation of the unified workload capacity space by extending the transformations of the parallel model bounds to the general case of n channels, where $n \geq 2$. We also provide the alternative versions for the unified space when the marginal distributions of the channels are assumed to be independent and identically distributed (IID), which serves to simplify the computations. Finally, in addition to the AND and OR cases derived in previous work, we add the bounds for single-target self-terminating processing, recently introduced in Blaha (2010) and Blaha & Townsend (under review).

We use the following notation throughout the paper. Let $F_{\mathcal{C}}(t) = P\left[T_{\mathcal{C}} \leq t\right]$ be the CDF of response times for a system with the set of n active channels, $\mathcal{C} = \{1, \ldots, n\}$. To denote the CDF of a single channel c among the $\mathcal{C}$ channels, we use $F_{c,\mathcal{C}}(t)$, and to denote the processing of a single channel c alone (i.e. no other active channels in the model or $n = 1$), we use $F_c(t)$. We use set minus notation $\mathcal{C} \setminus \{c\}$ to indicate the full set of channels $\mathcal{C}$ except c.

In this work, standard parallel processing is used to refer to a processing system that exhibits independent channel distributions (no cross-talk, no statistical facilitation/degradation). This means that for any number of active channels, the CDF for all channels active simultaneously is the product of the marginal distributions, $F_{\mathcal{C}}(t) = \prod_{c=1}^{n} F_{c,\mathcal{C}}(t)$. Additionally, standard parallel processing exhibits context independence, or context invariance. This means that the marginal distribution of any given channel c is identically distributed when any number of additional channels are also operating. We denote this by $F_c(t) = F_{c,\mathcal{C}}(t)$. Functionally, this allows the individual channels to be estimated by single-target or single-feature conditions in an experiment, which often greatly simplifies the number of conditions the experimenter needs to test in order to use these models.

Additionally, we note that standard parallel processing is often referred to as the parallel race model, the parallel horse-race model, or simply the race model (see, for example, Miller, 1982). This analogy specifically refers to the case when the fist channel to finish processing is enough to make a response. This is the case of minimum time processing, also termed first-terminating stopping or an OR (logical OR-gate) stopping rule. This would be the stopping rule engaged in tasks like visual search among redundant targets (no distractors) where the identification of the first target to be searched is enough to complete the task. The standard parallel model architecture is engaged under other stopping rules, as well, including exhaustive stopping (last-terminating or logical AND stopping), and the in-between case of single-target self-terminating (ST-ST) stopping. In the former case, all channels must complete processing before a response is made. In the latter case, the completion of a specific single target channel is enough to terminate the processing, but the target channel may be any of the n possible channels - first, last, or somewhere in between. Each of these stopping rules changes the form of the capacity coefficient and the predictions of the race model bounds, so we will present the derivation

of the bounds in unified workload capacity space for each model in turn.

Before we get into the derivation of the unified response time bounds, we want to remind readers that all CDFs and survivor functions exist on the range $[0, 1]$, so the natural logarithm of those functions produce negative values. Thus, cumulative reverse hazard functions (natural log of the CDF) exist on the range $(-\infty, 0]$, as do the natural logarithms of any bounds formed by a single CDF or products of CDFs (sums of CDFs can range above 1, and so the natural log can exist on $(-\infty, +\infty)$). These negative values will influence the derivation of inequality chains throughout this paper. Note also that the cumulative hazard function used in the minimum time bounds, is found as the *negative* natural log of the survivor function, and so it exists on the range $[0, +\infty)$, leaving fewer negative signs to track in those proofs.

Minimum Time Bounds

Let $F_{\mathcal{C}}(t) = P\left[\min_{\mathcal{C}}(T_c) \leq t\right]$, for all real $t \geq 0$ and $c \subset C$, be the CDF for an n-channel system operating under a minimum time stopping rule, where $\mathcal{C} = \{1, \ldots, n\}$ is the set of all possible channels. Define $F_{\mathcal{C}\setminus\{i\}}(t) = P\left[\min_{\mathcal{C}\setminus\{i\}} T_c \leq t\right]$ as the CDF if all channels except i are running, and define $F_{\mathcal{C}\setminus\{i,j\}}(t) = P\left[\min_{\mathcal{C}\setminus\{i,j\}} T_c \leq t\right]$, $i \neq j$, for the CDF of all channels but i and j. Further, define the survivor function as $S_{\mathcal{C}}(t) = 1 - F_{\mathcal{C}}(t)$.

We measure the amount of work completed in each channel c with the cumulative hazard function, defined as:

$$H_c(t) = \int_{\tau=0}^{t} \frac{f_c(\tau)}{S_c(\tau)} d\tau = -\ln\left(S_c(t)\right)$$

which can easily be estimated directly from the empirical response time survivor function for any experimental condition.

The capacity coefficient for minimum time (first-terminating, OR) processing for an n-channel model is defined as a ratio of cumulative hazard functions (Townsend &

Nozawa, 1995; Townsend & Wenger, 2004):

$$C_{\mathrm{OR}}(t) = \frac{H_{\mathcal{C}}(t)}{\sum_{c=1}^{n} H_c(t)}. \tag{1}$$

The numerator in Equation 1 is the observed processing of n active channels, while the denominator is the prediction of a benchmark standard parallel processing model, exhibiting independence and, in this terminology, unlimited capacity. Thus, capacity is qualitatively inferred relative to a ratio equal to 1, which is where observed processing is equal to the benchmark model prediction and unlimited capacity is concluded. If $C_{\mathrm{OR}}(t) > 1$, then super capacity, or better-than-benchmark, performance is inferred. And if $C_{\mathrm{OR}}(t) < 1$, then limited capacity, worse-than-benchmark performance, is inferred.[1]

The original race model CDF bound by Miller (1982) provided an upper bound on the CDF from the parallel, minimum-time model with $n = 2$ channels given by $F_{\{A,B\}}(t) \leq F_A(t) + F_B(t)$, where A, B denote the two parallel channels. Grice et al. (1984) introduced the concept of a lower bound for the same processing model, which is defined as $F_{\{A,B\}}(t) \geq \min\left[F_A(t), F_B(t)\right]$.

Colonius & Vorberg (1994) provided the n-channel generalization of both CDF bounds on parallel minimum-time processing in the inequality chain

$$\max_{i}\left[F_{\mathcal{C}\setminus\{i\}}(t)\right] \leq F_{\mathcal{C}}(t) \leq \min_{i,j}\left[F_{\mathcal{C}\setminus\{i\}}(t) + F_{\mathcal{C}\setminus\{j\}}(t) - F_{\mathcal{C}\setminus\{i,j\}}(t)\right]. \tag{2}$$

Theorem 1. *The unified workload capacity space inequality chain for the capacity of an n-channel, minimum-time system is, for $i \neq j$,*

$$\frac{\ln\{\min_i\left[S_{\mathcal{C}\setminus\{i\}}(t)\right]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}} < C_{OR}(t) < \frac{\ln\{\max_{i,j}\left[S_{\mathcal{C}\setminus\{i\}}(t) + S_{\mathcal{C}\setminus\{j\}}(t) - S_{\mathcal{C}\setminus\{i,j\}}(t)\right]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}} \tag{3}$$

Proof. From Equation 1, $C_{\mathrm{OR}}(t) * \ln\{\prod_{c=1}^{n} S_c(t)\} = \ln\{S_{\mathcal{C}}(t)\}$. Rewrite the upper bound from Equation 2 in terms of the survivor functions to get

$$S_{\mathcal{C}}(t) \geq \max_{i,j}\left[S_{\mathcal{C}\setminus\{i\}}(t) - S_{\mathcal{C}\setminus\{j\}}(t) - S_{\mathcal{C}\setminus\{i,j\}}(t)\right].$$

It follows that

$$C_{\mathrm{OR}}(t) * \ln\{\prod_{c-1}^{n} S_c(t)\} \geq \ln\{\max_{i,j}\left[S_{\mathcal{C}\setminus\{i\}}(t) + S_{\mathcal{C}\setminus\{j\}}(t) - S_{\mathcal{C}\setminus\{i,j\}}(t)\right]\}$$
$$\Rightarrow C_{\mathrm{OR}}(t) \leq \frac{\ln\{\max_{i,j}\left[S_{\mathcal{C}\setminus\{j}(t) + S_{\mathcal{C}\setminus\{j}(t) - S_{\mathcal{C}\setminus\{i,j\}}(t)\right]\}}{\ln\{\prod_{c-1}^{n} S_c(t)\}}.$$

Similarly, rewrite the lower bound of Equation 2 as

$$\min_{i}\left[S_{\mathcal{C}\setminus\{i\}}(t)\right] > S_{\mathcal{C}}(t)$$

and it follows that

$$C_{\mathrm{OR}}(t) * \ln\{\prod_{c=1}^{n} S_c(t)\} \leq \ln\{\min_{i}\left[S_{\mathcal{C}\setminus\{i\}}(t)\right]\}$$
$$\Rightarrow C_{\mathrm{OR}}(t) \geq \frac{\ln\{\min_{i}\left[S_{\mathcal{C}\setminus\{i\}}(t)\right]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}.$$

□

Under the assumption that the marginal distributions for each channel are IID, then all $F_{\mathcal{C}\setminus\{i\}}(t)$ are the same for any choice of $i \in \mathcal{C}$ and we can write this as $F_{\mathcal{C}\setminus,1\}}(t)$ to denote the CDF for n 1 active channels. Similarly, the IID assumption means the $F_{\mathcal{C}\setminus\{i,j\}}(t)$ are the same for any choice of $i, j \in \mathcal{C}$, and we write this as $F_{\mathcal{C}\setminus\{1,2\}}(t)$ for the CDF with $n-2$ active channels. Consequently, Equation 2 simplifies to (Colonius & Vorberg, 1994)

$$F_{\mathcal{C}\setminus,1\}}(t) \leq F_{\mathcal{C}}(t) \leq \left[2 * F_{\mathcal{C}\setminus\{1\}}(t) - F_{\mathcal{C}\setminus\{1,2,\}}(t)\right]. \tag{4}$$

Lemma 1. *When the marginal distributions of the parallel model are IID, the unified workload capacity space inequality chain for the capacity of an n-channel, minimum-time system is defined by*

$$\frac{\ln\{S_{\mathcal{C}\setminus\{1}(t)\}}{\ln\{\prod_{c-1}^{n} S_c(t)\}} \leq C_{OR}(t) \leq \frac{\ln\{2 * S_{\mathcal{C}\setminus\{1\}}(t) - S_{\mathcal{C}\setminus\{1,2}(t)\}}{\ln\{\prod_{c-1}^{n} S_c(t)\}} \tag{5}$$

The proof of Lemma 1 is similar to the proof of Theorem 1 and is left to the reader.

Maximum Time Bounds

Let $G_{\mathcal{C}}(t) \quad P\left[max_{\mathcal{C}}(T_c) \leq t\right]$, where again $\mathcal{C} \quad \{1, \ldots, n\}$ is the set of all n channels and $c \subseteq \mathcal{C}$, be the cumulative distribution function of response times for an n-channel system under a maximum time (logical AND, exhaustive) stopping rule.[2]

In order for the capacity coefficient inferences to be consistent with those for Equation 1, we utilize the cumulative reverse hazard function to measure the work throughput for each channel in under the maximum-time stopping rule (for a full discussion of the reasoning, see Townsend & Wenger, 2004; Townsend & Eidels, 2011). The cumulative reverse hazard function for processing channel c is given by

$$K_c(t) = \int_{\tau=0}^{t} \frac{g_c(\tau)}{G_c(\tau)} d\tau = \ln\left(G_c(t)\right)$$

which, again, can easily be estimated directly from the empirical response time CDF for any experimental condition.

The capacity coefficient for maximum time processing is defined as (Townsend & Wenger, 2004)

$$C_{\text{AND}}(t) = \frac{\sum_{c=1}^{n} K_c(t)}{K_{\mathcal{C}}(t)} \tag{6}$$

The numerator in the AND case is the prediction of the benchmark unlimited capacity, independent parallel model, while the denominator is the observed processing of n channels under the maximum-time stopping rule. Capacity inferences, again, are relative to the value $C_{\text{AND}}(t) = 1$, which indicates unlimited capacity. $C_{\text{AND}}(t) > 1$ indicates super capacity processing, and $C_{\text{AND}}(t) < 1$ indicates limited capacity processing.

Derived by Colonius & Vorberg (1994), the general bounds for n exhaustively processed channels are

$$\max_{i,j}\left[G_{\mathcal{C}\setminus\{i\}}(t) + G_{\mathcal{C}\setminus\{j\}}(t) - G_{\mathcal{C}\setminus\{i,j\}}(t)\right] \leq G_{\mathcal{C}}(t) \leq \min_{i}\left[G_{\mathcal{C}\setminus\{i\}}(t)\right]. \tag{7}$$

Theorem 2. *The unified workload capacity space inequality chain for the capacity of an n-channel, maximum-time system is, for $i \neq j$,*

$$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\max_{i,j}\left[G_{\mathcal{C}\setminus\{i\}}(t) + G_{\mathcal{C}\setminus\{j\}}(t) - G_{\mathcal{C}\setminus\{i,j\}}(t)\right]\}} \leq C_{AND}(t) \leq \frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\min_i\left[G_{\mathcal{C}\setminus\{i\}}(t)\right]\}}. \quad (8)$$

Proof. From Equation 6, $C_{\text{AND}}(t) * \ln\{G_{\mathcal{C}}(t)\} = \ln\{\prod_{c=1}^{n} G_c(t)\}$. Utilizing Equation 7, it follows that, for the upper bound

$$\begin{aligned} C_{\text{AND}}(t) * \ln\{G_{\mathcal{C}}(t)\} &\leq C_{\text{AND}}(t) * \ln\{\min_i\left[G_{\mathcal{C}\setminus\{i\}}(t)\right]\} \\ \Rightarrow \ln\{\prod_{c=1}^{n} G_c(t)\} &\leq C_{\text{AND}}(t) * \ln\{\min_i\left[G_{\mathcal{C}\setminus\{i\}}(t)\right]\} \\ \therefore \frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\min_i\left[G_{\mathcal{C}\setminus\{i\}}(t)\right]\}} &\geq C_{\text{AND}}(t). \end{aligned}$$

Similarly, for the lower bound,

$$\begin{aligned} C_{\text{AND}}(t) * \ln\{G_{\mathcal{C}}(t)\} &\geq C_{\text{AND}}(t) * \ln\{\max_{i,j}\left[G_{\mathcal{C}\setminus\{i\}}(t) + G_{\mathcal{C}\setminus\{j\}}(t) - G_{\mathcal{C}\setminus\{i,j\}}(t)\right]\} \\ \therefore \ln\{\prod_{c=1}^{n} G_c(t)\} &\geq C_{\text{AND}}(t) * \ln\{\max_{i,j}\left[G_{\mathcal{C}\setminus\{i\}}(t) + G_{\mathcal{C}\setminus\{j\}}(t) - G_{\mathcal{C}\setminus\{i,j\}}(t)\right]\} \\ \Rightarrow \frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\max_{i,j}\left[G_{\mathcal{C}\setminus\{i\}}(t) + G_{\mathcal{C}\setminus\{j\}}(t) - G_{\mathcal{C}\setminus\{i,j\}}(t)\right]\}} &\leq C_{\text{AND}}(t). \end{aligned}$$

□

Under the assumption that the marginal distributions for each channel are IID, for any choice of $i \in \mathcal{C}$, all $G_{\mathcal{C}\setminus\{i\}}(t)$ are the same and for any choice of $i, j \in \mathcal{C}$, all $G_{\mathcal{C}\setminus\{i,j\}}(t)$ are the same. We write these as $G_{\mathcal{C}\setminus\{1\}}(t)$ and $G_{\mathcal{C}\setminus\{1,2\}}(t)$, for $n-1$ and $n-2$ active processing channel systems, respectively. It follows that Equation 7 simplifies to (Colonius & Vorberg, 1994):

$$2 * G_{\mathcal{C}\setminus\{1\}}(t) - G_{\mathcal{C}\setminus\{1,2\}}(t)] \leq G_{\mathcal{C}}(t) \leq G_{\mathcal{C}\setminus\{1\}}t). \quad (9)$$

Lemma 2. *When the marginal distributions of the parallel model are IID, the unified workload capacity space inequality chain for the capacity of an n-channel, maximum-time system is defined by*

$$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{2 * G_{\mathcal{C}\setminus\{1\}}(t) - G_{\mathcal{C}\setminus\{1,2\}}(t)\}} < C_{AND}(t) < \frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{G_{\mathcal{C}\setminus\{1\}}(t)\}}. \tag{10}$$

The proof of Lemma 2 is similar to the proof of Theorem 2 and is left to the reader.

Single-Target Self-Terminating Bounds

Blaha (2010) recently introduced a new capacity coefficient for ST-ST processing, with full details explicated in Blaha & Townsend (under review). For completeness with respect to the results in Townsend & Eidels (2011), we here give the ST-ST parallel processing CDF bounds for both $n = 2$-channel models and $n \geq 2$-channel models.

Let $F_{k,\mathcal{C}}(t) = P\left[T_{k,\mathcal{C}} < t\right]$ denote the CDF of response times for target channel $k \in \mathcal{C}$. Let $K_{k,\mathcal{C}}(t) = \int_{\tau=0}^{t} \frac{f_{k,\mathcal{C}}(\tau)}{F_{k,\mathcal{C}}(t)} d\tau = \ln(F_{k,\mathcal{C}}(t))$ be the cumulative reverse hazard function for target channel $k \in \mathcal{C}$.

The capacity coefficient for ST-ST processing is defined as (Blaha, 2010; Blaha & Townsend, under review)

$$C_{\text{STST}}(t) = \frac{K_k(t)}{K_{k,\mathcal{C}}(t)} \tag{11}$$

The benchmark parallel model is in the numerator of $C_{\text{STST}}(t)$, and the observed processing of target channel k among n active channels is in the denominator. The inferences about unlimited, limited, and super capacity are the same as the OR and AND models.

The bounds on ST-ST processing are

$$\prod_{c=1}^{n} F_c(t) < F_{k,\mathcal{C}}(t) < \sum_{c=1}^{n} F_c(t). \tag{12}$$

For $n = 2$ channels, with the two channels denoted $C = \{1, 2\}$, the bounds simplify to

$$F_1(t) * F_2(t) \leq F_{k,C}(t) \leq F_1(t) - F_2(t).$$

Theorem 3. *The unified workload capacity space inequality cham for the capacity of an n-channel, single-target self-terminating system is,*

$$\frac{\ln\{F_k(t)\}}{\sum_{c=1}^{n} \ln\{F_c(t)\}} \leq C_{STST}(t) \leq \frac{\ln\{F_k(t)\}}{\ln\{\sum_{c=1}^{n} F_c(t)\}}. \tag{13}$$

The proof of Theorem 3 is nearly identical to the proof of Theorem 2, substituting the capacity coefficient and bounds for ST-ST capacity in for those of maximum-time processing.

Under the assumption that the marginal distributions for each channel are IID, we use the CDF of a single channel $c \in C$, and rewrite Equation 12 as

$$[F_c(t)]^n \leq F_{k,C}(t) \leq n * F_c(t). \tag{14}$$

Lemma 3. *When the marginal distributions are IID, the unified capacity space bounds for ST-ST processing are*

$$\frac{\ln\{F_k(t)\}}{n * \ln\{F_c(t)\}} \leq C_{STST}(t) \leq \frac{\ln\{F_k(t)\}}{\ln\{n * F_c(t)\}}. \tag{15}$$

The proof of this is trivial and left to the reader.

Conclusion

We have provided the straight-forward extension of the unified workload capacity space bounds for standard parallel processing from the limited existing definitions for $n \quad 2$ channels given in Townsend & Eidels (2011) to the full $n \geq 2$-channel situation for

minimum-time, maximum-time, and single-target self-terminating stopping rules. The full set of bounds, including all special cases considered to date, are summarized in Table 1. This extension enables powerful generalizations of this approach to multiple stopping rules and any number of channels of interest, in order to model the complete processing mechanisms for an experiment of interest. Mapping the bounds onto the unified capacity space for any number of channels enables a single plot to be used to compare the capacity coefficient values to the upper and lower bounds on standard parallel processing in order to make more direct inferences about extreme capacity values.

References

Blaha, L. M. (2010). *A dynamic hebbian-style model of configural learning.* Unpublished doctoral dissertation, Indiana University, Bloomington, Indiana.

Blaha, L. M., & Townsend, J. T. (under review). On the capacity of single-target self-terminating processing.

Colonius, H., & Vorberg, D. (1994). Distribution inequalities for parallel models with unlimited capacity. *Journal of Mathematical Psychology, 38*, 35–58.

Grice, G. R., Canham, L., & Gwynne, J. W. (1984). Absence of a redundant-signals effect in a reaction time task with divided attention. *Perception & Psychophysics, 36*, 565–570.

Houpt, J. W., & Townsend, J. T. (2012). Statistical measures for workload capacity analysis. *Journal of Mathematical Psychology, 56*, 341–355.

Miller, J. (1982). Divided attention: Evidence for coactivation with redundant signals. *Cognitive Psychology, 14*, 247–279.

Townsend, J. T., & Eidels, A. (2011). Workload capacity spaces: A unified methodology for response time measures of efficiency as workload is varied. *Psychonomic Bulletin & Review*, *18*, 659–681.

Townsend, J. T., & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology*, *39*, 321–360.

Townsend, J. T., & Wenger, M. J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review*, *111*, 1003–1035.

Author Note

Correspondence should be addressed to Leslie Blaha, 711 HPW/RHCV, Wright-Patterson AFB, Ohio, 45433, Leslie.Blaha@us.af.mil.

This research was supported by AFOSR grant 12RH14COR to L.M.B and AFOSR FA9550-13-1-0087 to J.W.H.

Footnotes

[1]We note that appropriate statistical tests for inferences about $C_{\mathrm{OR}}(t)$ are available (Houpt & Townsend, 2012), but their details are beyond the scope of this paper.

[2]Note that the change in notation here is to simply help the reader distinguish the CDFs for minimum- and maximum-time stopping rules.

Table 1

Summary of all Bounds on the Capacity Coefficient

LOWER BOUNDS

Stopping Rule	n-channels	n IID channels	2 channels
OR	$\frac{\ln\{\min_i[S_{\mathcal{C}\setminus\{i\}}(t)]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{S_{\mathcal{C}\setminus\{1\}}(t)\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{\min[S_1(t),S_2(t)]\}}{\ln\{S_1(t)*S_2(t)\}}$
STST	$\frac{\ln\{F_k(t)\}}{\sum_{c=1}^{n}\ln\{F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{n*\ln\{F_1(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{F_1(t)*F_2(t)\}}$
AND	$\frac{\ln\left(\prod_{c=1}^{n} G_c(t)\right)}{\ln\{\max_{i,j}[G_{\mathcal{C}\setminus\{i\}}(t)\ G_{\mathcal{C}\setminus\{j\}}(t)\ \ G_{\mathcal{C}\setminus\{i,j\}}(t)]\}}$	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{2*G_{\mathcal{C}\setminus\{1\}}(t)-G_{\mathcal{C}\setminus\{1,2\}}(t)\}}$	$\frac{\ln\{G_1(t)*G_2(t)\}}{\ln\{G_1(t)+G_2(t)-1\}}$

UPPER BOUNDS

Stopping Rule	n-channels	n IID channels	2 channels
OR	$\frac{\ln\{\max_{i,j}[S_{\mathcal{C}\setminus\{i\}}(t)+S_{\mathcal{C}\setminus\{j\}}(t)-S_{\mathcal{C}\setminus\{i,j\}}(t)]\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{2*S_{\mathcal{C}\setminus\{1\}}(t)\ \ S_{\mathcal{C}\setminus\{1,2\}}(t)\}}{\ln\{\prod_{c=1}^{n} S_c(t)\}}$	$\frac{\ln\{S_1(t)+S_2(t)-1\}}{\ln\{S_1(t)*S_2(t)\}}$
STST	$\frac{\ln\{F_k(t)\}}{\ln\{\sum_{c=1}^{n} F_c(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{n*F_1(t)\}}$	$\frac{\ln\{F_k(t)\}}{\ln\{F_1(t)+F_2(t)\}}$
AND	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{\min_i[G_{\mathcal{C}\setminus\{i\}}(t)]\}}$	$\frac{\ln\{\prod_{c=1}^{n} G_c(t)\}}{\ln\{G_{\mathcal{C}\setminus\{1\}}(t)\}}$	$\frac{\ln\{G_1(t)*G_2(t)\}}{\ln\{\min[G_1(t),G_2(t)]\}}$

The Points to Pixels Pipeline (P2P^2): an Open Source Framework for Multivariate, Similarity, and Network Data Visualization*

Dustin Arendt*‡ Air Force Research Laboratory

Brett Jefferson Indiana University

Simon Su Air Force Research Laboratory

ABSTRACT

Principal Components Analysis, Multidimensional Scaling, and other advanced dimension reduction techniques are often used to to help visualize complex multivariate datasets. However, these visualizations reduce observations to a cloud of points, which may stop short of conveying more the interesting topological relationships present. Our contribution is P2P^2, a modular framework for transforming a set of points or observations into a visualization that conveys information about the simplicial complexes present in the dataset. The framework is abstracted in a manner that open source Python packages are be leveraged to perform the computational "heavy lifting." In addition to making this framework accessible to a much wider audience, this allows a more sophisticated component to replace nearly any portion of the pipeline. An additional contribution of this work is a robust method for computing a global distance threshold that is grounded in information theory and complex network theory.

Index Terms: G.2.2 [Discrete Mathematics]: Graph Theory—Graph Algorithms; H.1.2 [Information Systems]: User/Machine Systems—Human information processing;

1 INTRODUCTION

MANY problems in data analytics revolve around discovering useful insights from a set of observations. However, observations could take on many different forms given the context. For example, observations could consist of sets of points embedded in high dimensional space. Or, observations could be measured as a matrix of similarities between other observations. Furthermore, observations could be represented as a network of relationships between other observations. Additionally, each observation might also be labeled with some kind of descriptive attribute or class (e.g., man/woman, young/old, sick/healthy, etc.).

Given a set of observations, useful insights we may wish to gain can be answers to questions like:

- Which observations are "normal" and which are "outliers?"
- Do observations group together, and how are those groups related? and
- What is the relationship between classes and observations?

There are many sophisticated techniques from machine learning, statistics, complex network analysis, and other fields that can be applied to help answer the above questions. However it is usually the case that specific criteria about the data and the answer being sought must be met before any technique can be effectively leveraged. For this reason, visual analytics can be employed to obtain an overview or basic intuition about a dataset before more sophisticated techniques are applied.

*PA#

†National Research Council Resident Research Associate

‡e-mail: dustin.arendt.ctr@us.af.mil

Many of the popular tools for dimension reduction and embedding assume the problem is solved once the set of observations have been suitably mapped into the desired lower dimension. Furthermore, many algorithms (e.g,. Isomap, Locally Linear Embedding, Spectral Embedding) rely on determining the nearest neighbors for observations, which induces a graph from the set of observations. However, this graph, though it may contain useful information about the underlying topology of the dataset, is not often represented in the visualization. One contribution of P2P^2 is that it goes beyond simply determining a good placement for each vertex; P2P^2 also determines how to appropriately fill in the space *in between* points based on the topology of the underlying observations.

Towards this end, this paper presents a general methodology for data visualization that can be applied broadly to many of the different types of observations described above. Recently there has a been significant improvements in the availability, quality, and ease of use of open source tools for scientific computing and data analysis, especially for the Python scripting language. In this paper we also highlight how several open source packages can be combined to perform nearly all of the "heavy lifting" required to implement this pipeline. Aside from making the proposed visualization technique accessible to a wider audience, reliance on open source software in this manner abstracts the visualization pipeline in a way that is easily extensible. It is straightforward to plug in a different algorithm at the user's discretion for any of the main steps in the pipeline.

2 BACKGROUND & RELATED WORK

The problem of how to best display a set of (possibly high dimensional) observations in a low dimensional space is so fundamental to understanding scientific datasets that the basic techniques have been used for decades. For this type of problem we have a dataset X consisting of n p-dimensional observations, where $x_i \in \mathbb{R}^p$. One such technique is Principal Components Analysis (PCA), where a set of high dimensional points are rotated in a manner that gives the leading components the most amount of variance [9]. PCA can be used as a data visualization and exploration tool; when the number of principal components is 2 or 3, the points can be rendered on the screen to reveal relationships within the data. PCA can also be used simply as a dimension reduction technique for preprocessing before the data is tackled by other algorithms. Multidimensional scaling (MDS) addresses a problem similar to PCA, but assumes that only the distances between points are known [3]. In other words, the datasets D consists of n observations of n dimensions where d_{ij} is the observed distance between observation i and j. Such data could arise from preference questionnaires, for example. Essentially, MDS algorithms attempt to minimize the discrepancy between the distances separating the embedded points and the distances given in the input.

Many other techniques exist that improve on PCA or MDS in some way, with one of the most popular techniques being Isomap [17]. This algorithm constructs a graph connecting the closest observations to each other, and then the geodesic distances between these neighboring observations is used to determine a suitable lower dimensional embedding. Other embedding techniques include Locally Linear Embedding [14], Laplacian Eigenmaps [2], and Spec-

tral Embedding [11]. All of the techniques discussed above are available in the *Scikit-learn* python package for machine learning [13].

Suppose that in addition to having a set of observations X, that we also have a corresponding label for each observation Y, where $y_i \in \mathbb{Z}$. Visual analytics can help us discover if there is any meaningful relationship between X and Y. When we wish to learn a function $f : X \mapsto Y$ that accurately predicts y_i from x_i, this is referred to as *supervised learning*, (specifically classification when Y is discrete) [15]. Since this is such a common task, some researchers have developed algorithms that take into account both X and Y when determining the embedding, with one well known example being t-SNE [18].

3 Methods & Results

Given one of the following:

- X, a set of multivariate observations, where $\vec{x}_i \in \mathbb{R}^n$,
- D a matrix of dissimilarities between observations, or
- $G = (V, E)$ a graph representing the relationships between observations,

one of our goals is to determine an effective two-dimensional embedding of the observations, P, where p_i is the (x, y) coordinate of observation i in the visualization. Our approach, P2P^2, builds on the realization that several related modern algorithms compute the nearest neighbors of observations. Thus, there appears to be a very natural progression of $X \to D \to G \to P$ that will yield an effective visualization. Furthermore, at this level of abstraction, one can see that the same algorithm can be used regardless whether we start with X, D, or G–when we do not start with X, we are simply short-cutting the pipeline. Furthermore, each mapping in the pipeline is easily implemented with an open source Python package function call, making the mappings interchangeable with other algorithms according to one's preference.

However, we do not stop once P is computed, noting that G may have additional structure that can be conveyed effectively in a visualization. Specifically, G may contain a number of cliques (i.e., complete subgraphs), and we believe that rendering cliques as filled-in polygons instead of as the traditional node-link style improves the usability of the visualization. One reason for this belief is that a significant number of edges can be replaced with a single uniform polygon; a clique of size k would have be drawn as $k(k-1)$ edges, but would be replaced with a polygon having at most $k-1$ edges. The color of the polygon can encode the size of the clique so that density information can be gleaned from the visualization at a glance.

Given the set of maximal cliques C and an embedding of the observations P, it is necessary to compute the convex hull of each clique in order to render each clique as a polygon. This is because P actually defines a projection of the simplex corresponding to that clique into a lower dimensional space. As a result of this projection, some of the vertices in the simplex may end up as interior points. Computing the convex hull of the clique given P will identify which vertices are on the true boundary of the projected polygon.

The entirety P2P^2 procedure is described in Algorithm 1, and a description of open source function calls made by P2P^2 is shown in Table 1. The results of applying P2P^2 to are shown in Figures 1, 2, and 3. Figure 1 illustrates how computing the convex hulls of cliques might improve the usability of the visualization by reducing the number of edges drawn, and filling in some of the negative space.

Figure 2 shows how choosing the distance threshold ε affects the visualization. When ε is trivially small, no observations are considered neighbors, which results in a completely disconnected graph. As ε increases more of the underlying structure becomes evident, until eventually ε is large enough that all observations are considered neighbors and belong to a single clique. Clearly a value for ε that is "just right" must lie somewhere between these two extremes, which have little utility. We provide a way to compute a solution to this "Goldilocks" problem, which is detailed in Section 4.2.

Figure 3 shows P2P^2 applied to a larger multivariate dataset representing over 1000 hand drawn digits.

Algorithm 1 Basic outline of the P2P^2 abstraction

1: start with X, a $n \times m$ matrix of points
2: find (or start with) D, a $n \times n$ matrix of distances between points
3: find ε, a global distance threshold
4: find (or start with) G, a graph induced from D and ε
5: find P, an embedding of G, D, or X in $\mathbb{R}^2$
6: find C, the set of maximal cliques in G
7: find H, the convex hulls of each clique in C given P
8: draw each hull in H as a 2-D polygon

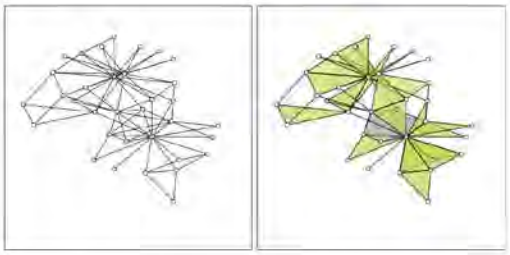

Figure 1: Comparison of a typical node-link rendering of a network (left) versus the P2P^2 embellishment (right). The graph data is from the Zachary's karate club network [19] which can be accessed with networkx.karate_club_graph.

4 Discussion

4.1 Considerations for Large Datasets

The $X \to D$ mapping will create scalability issues when $|X|$ is large; when the number of observations is becomes much greater than 10^4 the average machine will not have sufficient memory to store D, which grows as $O(|X|^2)$. To address this issue, one can instead map X directly to G by leveraging sophisticated data structures like the KDTree. For example, Scipy's KDTree allows for the efficient computation of k-ε neighbors (the k nearest neighbors with a distance less than ε) given a set of points, X. However, the KDTree can becomes inefficient when the number of dimensions are high (e.g., > 15 for this case). So when the dataset is both large and high dimensional, a reasonable solution to this problem is to use a fast dimension reduction technique like PCA to reduce the number of dimensions in X down to a tractable number, hopefully without much loss in accuracy.

4.2 Choosing ε

P2P^2 maps an X or D to a graph through the application of an arbitrary distance threshold. Clearly this threshold can have a significant detrimental effect to the usability of the visualization if chosen poorly, as demonstrated with Figure 2. So we outline here a method for determining a good choice for ε with a solid basis from information theory and complex network theory. First, we assume that we have a set of class labels Y corresponding to each observation in X or D. The intuition for choosing a good value of ε is that the edges in the graph induced by ε should have edges that are likely to connect observations with the same label. This is often referred to

Table 1: P2P^2 Open Source Function Calls

step	function call	description	ref
1,5	sklearn.decomposition.PCA	dimension reduction (use for large, high dimensional datasets) or as a vertex embedder	[9, 13]
2	scipy.spatial.distance.pdist	compute the distance between all pairs of points (many distance norms available to choose from)	[10]
3	scipy.optimization.minimize_scalar	scalar minimization of objective (use *bounded=True* with *bounds*$=(0,\varepsilon_{max})$–see Eqn. 7)	[4, 10]
4	scipy.spatial.KDTree	efficient computation of k-ε neighbors (use for large datasets, but only implemented for L_p norms)	[10]
4	networkx.Graph	create a graph data structure a list of edges (from nearest neighbors) or an adjacency matrix (from thresholded distance matrix)	[7]
5	networkx.draw_graphviz	compute 2-d spatial embedding for graph vertices (use "sfdp" option for very large graphs)	[6]
5	sklearn.manifold.MDS	embed vertices into $\mathbb{R}^2$ directly from D	[3, 13]
6	networkx.find_cliques	find maximally connected components (e.g., complete subgraphs)	[5, 7]
7	scipy.spatial.ConvexHull	"rubber band" fit to a set of points in order to omit interior points from the projection of the simplex	[1, 10]
8	matplotlib.collections.PolyCollection	render polygons	[8]

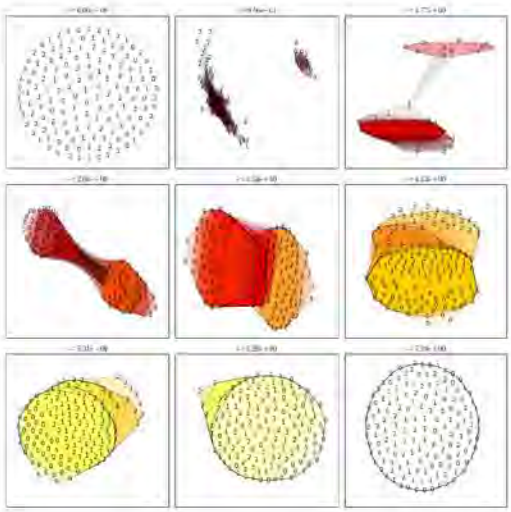

Figure 2: P2P^2 visualization of the Iris classification dataset with ε varying linearly from the minimum to maximum euclidean distance between observations. The data can be accessed with sklearn.datasets.load_iris.

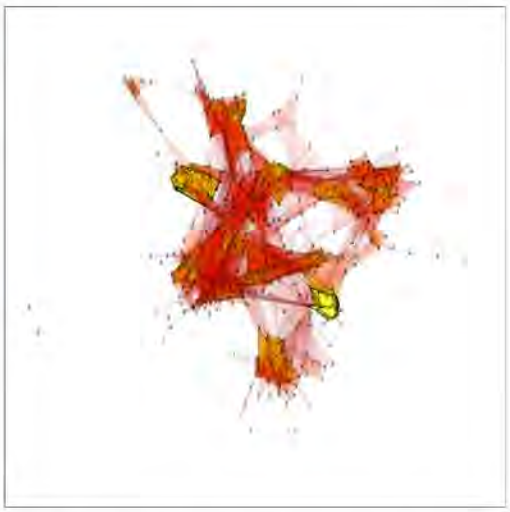

Figure 3: P2P^2 visualization of the Digits classification dataset. The digits dataset can be accessed with sklearn.datasets.load_digits.

as "assortative mixing," or the tendency for adjacent nodes in a network to have the same properties [12], and a number of techniques exist to measure it.

Classically, assortative mixing is measured with statistical correlational techniques, which has several known issues. Recently information theory, specifically mutual information has proven to be a useful tool for understanding complex networks, including the phenomenon of assortative mixing [16]. Thus, our measure for assortative mixing is based on the mutual information $I(E;A)$, where E and A are Boolean random variables corresponding to two nodes having equal state, and two nodes being adjacent, respectively. $I(E;A)$ is found as follows:

$$I(E;A) = H(E) - H(E|A), \tag{1}$$

and

$$H(E|A) = P(A=1)\cdot H(E|A=1) + P(A=0)\cdot H(E|A=0), \tag{2}$$

where H is the entropy of an arbitrary probability distribution X. The following probabilities are directly measured given $G=(V,E)$ and X, the network and the state of the nodes on the network, re-

spectively.

$$P(E=1) = \frac{1}{n}\sum_{i\in\Omega} c_i(c_i-1), \quad (3)$$

$$P(A=1) = \frac{2m}{n(n+1)}, \quad (4)$$

$$P(E=1|A=1) = \sum_{(u,v)\in E} \delta(y_u - y_v), \quad (5)$$

where $n=|V|$, $m=|E|$, $c_i=\sum_j \delta(y_j-i)$ (which simply counts the number of nodes in the graph with class i), Ω is the set of unique classes in Y, and δ is the Dirac delta function. The remaining quantity that is not immediately found due to the law of total probability is

$$P(E=1|A=0) = \frac{n[P(E=1)-P(E=1|A=1)]}{\frac{n(n-1)}{2}-m}. \quad (6)$$

Note that computing $I(E;A)$ scales effectively with the size of the dataset since, the most expensive computation loops over the edge set, requiring only $O(m)$ time to complete.

The ideal distance threshold is

$$\varepsilon_* = \arg\max_{\varepsilon} \mathscr{A}(\mathscr{G}(X,\varepsilon),Y),$$

where $\mathscr{G}(X,\varepsilon)$ induces a graph from the set of observations X and a given ε, and $\mathscr{A}(G,Y)$ computes the assortative mixing of the graph G with corresponding labels Y. The optimization program can be bounded to $(0,\varepsilon_{max})$ where

$$\varepsilon_{max} = \max_{ij} d(x_i,x_j). \quad (7)$$

If X is large (e.g., $|X|>10^3$) then a smaller random subset of X should be a sufficient replacement for X to compute ε_{max}.

One subtle issue is that because mutual information must be positive, then both $H(A)$ and $H(E)$ are upper bounds of $I(E;A)$. As we vary ε, we can expect $H(E)$ to remain constant; however as we increase ε, the density of the graph also increases, causing $H(A)$ to increase until the density of the graph reaches 1/2. Therefore, a fairer way to compare the mixing of two graphs that have different densities would be to normalize the mutual information by $H(A)$. Figure 4 shows the beneficial effect this normalization has on the mixing scores for the digits dataset. When $I(E;A)$ is normalized by $H(A)$, the optimal value of ε decreases.

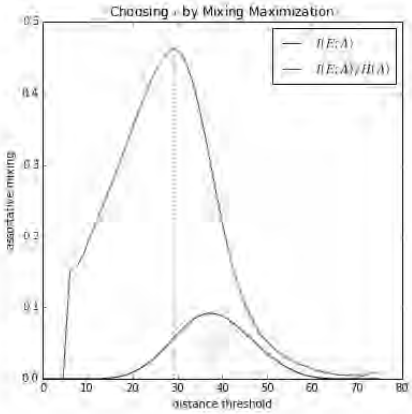

Figure 4: Mutual information scores of the graph induced by ε for the digits dataset. Normalizing $I(E;A)$ by $H(A)$ decreases the optimal value of ε.

5 Conclusions & Future Work

Conclusion

For future work we intend to implement P2P^2 in an immersive 3-D virtual environment. From an algorithmic standpoint, making this jump is nearly trivial due to the modular nature of P2P^2. One must simply exchange the current graph drawing algorithm with one that embeds vertices into 3-D (or, alternatively use MDS or PCA to project X or D into 3 instead of 2 dimensions). Furthermore, the convex hull algorithm also generalizes to 3 dimensions, where polygons are simply sets of triangles instead of line segments. The main effort in the implementation in the virtual environment will be in an effective interface for exploratory visual analytics, as well as effective volume rendering of the 3-d convex hulls.

Acknowledgements

TODO

References

[1] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[3] I. Borg and P. J. F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, 2005.

[4] R. P. Brent. *Algorithms for minimization without derivatives*. Courier Dover Publications, 1973.

[5] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[6] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz–open source graph drawing tools. *Graph Drawing*, pages 483—484, 2002.

[7] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, Aug. 2008.

[8] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[9] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[10] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[11] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001.

[12] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[15] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[16] R. V. Solé and S. Valverde. Information theory of complex networks: On evolution and architectural constraints. In *Complex Networks*, pages 189–207. Springer, 2004.

[17] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[18] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(85):2579–2605, 2008.

[19] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473. 1977.

SIMCog-JS: Simplified Interfacing for Modeling Cognition - JavaScript

Tim Halverson (thalverson@gmail.com)
Oregon Research in Cognitive Applications, LLC
Oregon City, OR 97045 USA

Brad Reynolds (reynolds.157@wright.edu)
College of Engineering and Computer Science, Wright State University
Dayton, OH 45435 USA

Leslie Blaha (leslie.blaha@us.af.mil)
711th Human Performance Wing, Air Force Research Laboratory
WPAFB, OH 45433 USA

Abstract

A continuing hurdle in the cognitive modeling of human-computer interaction is the difficulty with allowing models to interact with the same interfaces as the user. Multiple attempts have been made to add this functionality (e.g., Hope, Schoelles, & Gray, 2014) in limited domains. This paper presents a solution allowing models to interact with web browser-based software, while requiring little modification to the task code. Simplified Interfacing for Modeling Cognition - JavaScript (SIMCog-JS) allows the modeler to specify how elements in the interface are translated into ACT-R chunks, allows keyboard and mouse interaction with JavaScript code, and allows sending ACT-R commands from the external software (e.g., to add instructions). The benefits, drawbacks, and future functionality of SIMCog-JS are discussed.

Keywords: Cognitive Architectures; Task Interface; ACT-R; WebSockets; JSON; HTML; JavaScript; D3

Introduction

A substantial challenge with modeling human cognition is the presentation of task environments to the simulated human. Software re-implementation provides little scientific reward, yet modelers face this burden every time they utilize a new or modified task. The situation is further complicated if a modeler is studying human-computer interaction (HCI) with complex software in which users are engaging in ongoing, dynamic, and interleaved or multi-tasking behaviors. Because the focus of cognitive modeling in HCI is often either explaining or predicting performance differences between alternative interfaces, substantial research time is spent re-implementing multiple, complex interfaces; this effort is further multiplied if multiple cognitive architectures are used.

Although re-implementation within a modeling architecture framework can allow maximum control by the modeler, it introduces additional challenges: (a) Re-implementation increases the likelihood that the fidelity of the simulation is degraded by an imperfect porting of the user interface or task dynamics. (b) Iterative changes to the original software/task require additional efforts to integrate these changes into the model's task environment. (c) Task-simulation environments for cognitive architectures are sometimes written in programming languages not commonly used for building HCI interfaces (e.g., ACT-R uses Lisp; Anderson et al., 2004) and often provide limited facilities for building the task simulations. Thus, the process of re-implementation forces a trade-off between task fidelity and time savings. An alternative to re-implementation is to allow a model to communicate directly with a user interface that is external to the cognitive architecture. Previous research has attempted to solve this challenge, although in limited domains. Computer vision (CV) has been used to automatically extract relevant visual features from an existing computer interface (e.g., Halbrügge, 2013; St Amant, Riedl, Ritter, & Reifers, 2005). While CV solutions remove the burden of "translating" the interface to symbols understood by the architecture, they also reduce the control the modeler has on how the visual interfaces are specified. Additional control requires the modeler to customize the CV algorithms or specify screen element "templates" at the pixel level. Other solutions provide the ability for models to act within specialized environments, like games (e.g., Veksler, 2009) or robotics (e.g., Kennedy, Bugajska, Adams, Schultz, & Trafton, 2008). These solutions are incredibly useful but are limited to their specialized environments. Still other solutions provide a more general framework for interfacing models with external software by using interprocess communication protocols available in many programming languages (e.g., Büttner, 2010; Hope et al., 2014). The solution presented herein falls into this final category.

We present a solution to the challenge of communication between external task environments and cognitive architectures: Simplified Interfacing for Modeling Cognition - JavaScript (SIMCog-JS). Our approach supports communication between Java ACT-R (Salvucci, 2013) and HTML-/JavaScript-based software in a user-friendly manner. In the remainder of this article, we specify some design requirements, describe the functionality provided by SIMCog-JS, and provide an example of SIMCog-JS applied to a dynamic, multitasking experiment environment.

SIMCog-JS Design Requirements

SIMCog-JS is a technology that allows cognitive modelers to specify how visual information is extracted from external software, passes that information to ACT-R, and passes keyboard and mouse events back to the external software. The primary motivation for SIMCog-JS is rooted in a desire to

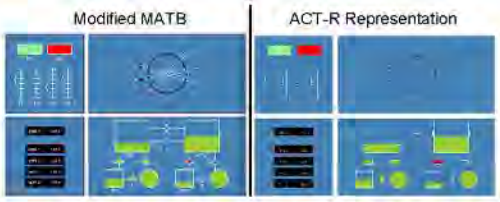

Figure 1: The browser-based modified Multi-Attribute Task Battery (mMATB) as it would appear to a human participant (left) and a representation of the ACT-R visicon (right).

apply cognitive architectures to dynamic, multitasking experiments, such as training simulations or naturalistic web-browsing. We desire a flexible system allowing interaction between multiple cognitive modeling formalisms and existing software/HCI environments.

SIMCog-JS attempts to minimize the modeler's burden in multiple ways. First, SIMCog-JS requires minimal modification of existing task code, although it does require that the modeler have access to the JavaScript task code. Second, SIMCog-JS includes an extension to Java ACT-R that replaces Java ACT-R task code and requires no modifications for a wide variety of tasks. Third, SIMCog-JS provides a user-friendly, flexible syntax for specifying which visual elements should be passed to ACT-R, when those elements should be updated in ACT-R, and how they should be perceived (i.e., slot values).

In order to provide this functionality, SIMCog-JS had three critical design requirements:

1. SIMCog-JS must use standard software protocols for communication between models and experimental software.
2. Integrating model interactions with the task makes minimal modifications to the experimental code, minimizing interference with human data collection or natural behaviors.
3. Model execution occurs in real time.[1]

We note that as our initial target task environment, the modified Multi-Attribute Task Battery (mMATB), executes in a web-browser and the modeling formalism, Java ACT-R, is written in Java, we were required to implement a new solution to facilitate interaction between cognitive models and a task environment. Hope et al. (2014) introduced a similar solution for interfacing Lisp ACT-R with stand-alone software. However, that published solution does not support either Java or JavaScript. Our solution took motivation from Hope et al.'s work.

The mMATB Task Environment

We apply SIMCog-JS to a dynamic, multitasking environment, mMATB (Cline, Arendt, Geiselman, & Blaha, 2014). This is a generalized version of the MATB developed to assess multitasking in pilot-like environments (Arnegard & Comstock, 1991); the modifications in this environment make similar cognitive demands on the participants, but the tasks are less pilot-specific in nature. Our browser-based implementation is written with the D3 JavaScript library (Bostock, Ogievetsky, & Heer, 2011) integrated with a Python django database. Participants interact with the environment through keyboard button presses and mouse clicks and movements.

The mMATB, shown in the left panel of Figure 1, entails four separate tasks, which we summarize clockwise from the upper left. The upper left quadrant is a Monitoring Task, consisting of a set of sliders and two color indicator blocks. The participant's task is to provide the appropriate button press (F1-F6, labeled on each indicator/slider) if a parameter is out of its normal state. For the sliders, this means moving above or below ± 1 notch from the center. For the indicators, the normally green (black) might turn black (red).

A Tracking Task is contained in the upper right quadrant, wherein three colored circles move continuously along individual ellipsoid trajectories. At any time, one of the circles may turn red, indicating it is the object to be tracked by the participant. The participant tracks the target by mousing to the target, clicking on it, and then following it with the mouse, until the next target object is indicated with a color change.

The lower right quadrant contains a Resource Management Task. Two resource tanks are schematically illustrated, together with representations of fuel sources, reserve tanks, and gated connections (each numbered 1-8) between all tanks. The participant's task is to maintain the resource levels within a range specified by bars on the sides of the tanks. The on/off states of the gates are controlled with number pad key presses. The participant can control the gates with any strategy of choice to maintain the resource levels.

Finally, the lower left quadrant contains a Communications Task. The display shows four channels (Int1, Int2, Ops1, Ops2) together with the current channel values; the topmost line gives a target channel and value. If a red cued target appears in the top box, the participant uses the up/down arrow keys to select the cued channel and the right/left arrow keys to adjust the channel value to the new cued value. The enter key submits the corrected channel, which changes the topmost cue box to white until the next channel cue appears.

Cognitive modeling of mMATB performance aims to capture behavioral impacts of changes in workload, operator stress levels, or fatigue levels and to characterize the high-level strategies engaged during continuous multitasking.

SIMCog-JS Software Architecture

SIMCog-JS uses a client-server software architecture. The server exists within Java ACT-R (Salvucci, 2013) as a "generic task." This generic task is populated with environment-specific information as the server receives messages from a client describing the current state of a task interface. The server dynamically changes the ACT-R envi-

[1] As our target software does not support synchronized execution with external software. This is not a constraint unique to our target software, as web browsers (and most software) do not allow external synchronization.

ronment based on the messages received from the client and sends messages to the client describing ACT-R's actions.

The client is built in JavaScript, allowing it to run within all modern web browsers. The client runs alongside the browser-based task translating the task interface for the server and processing interactions from ACT-R. The client is integrated into existing code by referencing the client script in the task's primary web page (e.g., index.html). Further, the modeler specifies three things within the client: (a) a list of visual chunks to be represented in ACT-R, (b) a list of ACT-R commands for the model, and (c) handlers for interactions received from the task.[2] Once these are in place, the system is ready for use.

The client and server communicate via WebSockets and JavaScript Object Notation-Remote Procedure Call (JSON-RPC). WebSockets (Fette & Melnikov, 2011) allow reliable, simultaneous connections between the client and server.[3] Once connected, the client and server use JSON-RPC (JSON-RPC Working Group, 2010) to send information. JSON-RPC is a standardized protocol for sending messages based on the JSON standard. Both the WebSocket and JSON-RPC protocols are standards that have been implemented in many programming languages, allowing SIMCog to be easily extended to task interfaces and cognitive modeling formalisms in other programming languages through the use of these standard protocols and reuse of the SIMCog-JS's messaging specification. WebSockets and JSON are native to JavaScript, but requires additional libraries for Java.

Figure 2 shows the flow of information between the browser task environment (i.e., client), the server, and ACT-R. After Java ACT-R and the client are configured and running, the client sends all information about the interface to the server at the start of the task, along with any initial ACT-R commands. As keyboard and mouse events are generated by ACT-R, these actions are passed to the client to affect the interface. Details on how to configure the client and server can be found with the SIMCog-JS Software Design Document included in SIMCog-JS distribution.[4] The following section provides details on how this communication takes place between the client and ACT-R.

Communicating through SIMCog-JS

SIMCog-JS allows the modeler to specify how interface elements in software will be represented in ACT-R, to send ACT-R commands from the task client to Java ACT-R, and to determine how the task client will respond to keypresses and cursor movements made by the model. The following sections describe how these three facilities are used.

[2]As discussed in Keypress and Mouse Events section below, default keypress and mouse click handlers are provided for the modeler's convenience.

[3]The client and server may be run on separate computers and over the internet. However, doing so may introduce additional lag that could reduce the fidelity of the simulation.

[4]The SIMCog-JS distribution can be downloaded from: http://sai.mindmodeling.org/simcog/

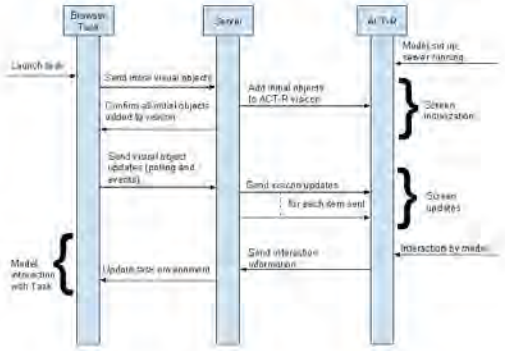

Figure 2: Information flow through SIMCog-JS. The information events start with the modeler initiating the Java ACT-R model and then launching the browser-based task. SIMCog-JS then connects the two environments. The vertical dimension captures time flowing from top to bottom.

Specifying Visual Chunks

The left side of Figure 1 shows the visual interface for mMATB task as presented to the human participant in the web browser. The right side of Figure 1 shows a visual representation of ACT-R's visicon, as specified by the modeler and displayed by the SIMCog-JS server. The modeler specifies which web-browser elements become visual chunks in ACT-R, how those elements will be represented in ACT-R, and when those elements will be updated. SIMCog-JS does not send all interface elements to ACT-R; doing so could unnecessarily complicate the modeling. The modeler may have observational data or theoretical reasons for hypothesizing that some interface elements are completely ignored by users. For example, a uniform background frame may have no impact on performance, assuming adequate contrast between the background and foreground elements. Therefore, the modeler must specify the set of interface elements that become visual chunks in ACT-R.

The modeler must specify the interface element id and the element's shape type. The object's coordinates, width, height, color, and text (if applicable) are automatically extracted from the task interface using JavaScript DOM function calls and jQuery-dependent CSS specificity computations.[5] The syntax for specifying visual objects is:[6]

[5]All attributes can be specified manually. See the Design Document at http://sai.mindmodeling.org/simcog/ for more information and useful links to the libraries utilized.

[6]All syntax descriptions follow the same convention. Angle brackets are used to indicate a value that must be specified by the modeler. Values enclosed in quotation marks indicate that the value is a string. For example, id:"<unique_name>" indicates that unique_name should be replaced by a string that is the value of the id, like id:"foo". Alternative values are separated by "|".

```
{id:"<unique name>", type:"<valid type>"
```

The id uniquely identifies the object. If the interface element has an explicitly labeled id in the document-object model (DOM; e.g.,), that string can be used as the SIMCog-JS id. If an object does not have a unique ID, the id can be specified with two attributes, name and domLocation. The name value is a string that must be unique to the object. It is helpful to make the name meaningful. A domLocation value is the node of the object located within the DOM tree. This node can be found in multiple ways. One way is to identify the relation to another named object within the DOM tree. Another is to locate the object in the DOM tree relative to the root (i.e., document). Syntax for these methods are:

```
{id:{domLocation:document.getElementById(
    "<element_id>").nextElementSibling,
  name:"<unique_name>"}, ...}

{id:{domLocation:document.body.
    lastElementChild,
  name:"<unique_name>"}, ...}
```

The type is a string that determines how the object will be represented within ACT-R. A screen object must be one of nine types: "Line", "Cross", "Label", "Oval", "OvalOutline", "OvalOutlineFill", "Rectangle", "RectangleOutline", or "RectangleOutlineFill". The first three types are "native" to Java ACT-R;[7] the remaining items are custom task components added by the authors. The type of an object is represented in the visual chunk's "isa" attribute (e.g., "OvalOutlineFill" has an attribute of "isa oval"). Additionally, if the shape is specified with two colors (e.g., "OvalOutlineFill" has a fill and outline color), then SIMCog-JS adds a borderColor chunk slot that contains the value of the border's color and the standard ACT-R color slot contains the value of the fill color. The coordinates, dimensions, and colors of objects are determined differently for different object types. If an object is declared with the wrong type, it is likely that the object will be misrepresented in ACT-R.

The modeler may also specify when changes to interface elements are sent to ACT-R. The default is to update whenever the element changes using DOM Mutation Observers. This event-based functionality is most useful when one or more attributes of the interface element changes infrequently. The modeler may also specify that updates occur at a configurable, regular interval (e.g., polling). This polling functionality is most useful when the attributes of objects are rapidly changing. In such cases, the polling method can substantially decrease the number of messages to the server, decreasing computational demands. Specifying polling-based changes is done by adding a change attribute with the value "poll" to the element declaration. Finally, an object can be declared as static. Static elements are never updated. The modeler may specify an object as static by adding a change attribute with the value "static" to the element declaration. Syntax for change declarations is:

```
{..., change:"evt"|"poll"|"static"}
```

In addition to specifying when updates for an object are sent, the modeler may specify which visual properties are updated. By default, all properties are updated. Listing only those properties that will change can improve software performance. For example, a light may only change color but not move, or tracking reticles may only change coordinates but not colors. The list of properties that will be updated are appended to the value given to the change attribute. If no such list is given, all properties are updated. Valid attributes are "x", "y", "height", "width", "color", "secondaryColor", and "stringVal". Only labels have "stringVal" attributes. Syntax of these expanded change declarations are:

```
{..., change:["<attribute_name>",
    "additional_attribute_name>, ...]}
{..., change:["poll", "<attribute_name>",
    "additional_attribute_name>, ...]}
```

It is also possible to add objects to the ACT-R task environment that are not relevant to the model but are useful for the modeler (i.e., for debugging the visual interface). This is done using "task-irrelevant" objects. Task-irrelevant objects never appear in the model's visicon. For example, a task-irrelevant object may be used as a background to make objects easier to see for the modeler. There are four possible task-irrelevant objects: Cross, Label, Line, and Rectangle. Task-irrelevant objects are not updated throughout the task. All objects default to being task-relevant. To declare an object as task-irrelevant, the attribute taskRelevant is added to an object declaration with a value of false. The syntax for this option is:

```
{..., taskRelevant:true|false}
```

Example Specifications from mMATB This section provides examples of how interface elements in the mMATB task, shown in Figure 1, are specified. The examples start with simple specifications and progress to the more complex.

Perhaps the simplest interface elements in mMATB are the background color panels underlying all four quadrants. They never change (i.e., are static), are filled with a single color ("steel blue"), and are rectangular. If one hypothesizes that these background colors are ignored by the users, these elements can be declared as task-irrelevant. Alternatively, the cognitive model could simply ignore these elements, or the modeler could choose to exclude these elements. Making them task-irrelevant will improve software performance ever so slightly. Including them in the interface specification will make the interface in ACT-R more readable. Although the interface element is simple, it is not uncommon for HTML ids to be missing from background elements, which complicates the id for these elements. In this example, the domLocation value is used to determine the id based on the modeler's knowledge of the location of these elements in the DOM tree.

[7] Note that "Button"s are not supported. As discussed later, any type of object can be clickable.

```
{type:"Rectangle",
  id:{name:"svg0",
    domLocation:d3.
      selectAll("svg")[0][0].firstChild},
  change:"static",
  taskRelevant:false}
```

The Monitoring Task color indicator blocks (upper left quadrant) provide a straightforward example for displaying event-based task elements. The following example is the specification of the green color indicator block; the specification of the red block is similar. The id of this rectangular element is known, monitor_button_0. The only property that changes is the color and so the only value assigned to the change attribute is color. The changes are infrequent, normally changing only a few times per second, so the change attribute is given the value of "evt". Note that "evt" is the default and is not required in the declaration.

```
{type:"Rectangle",
  id:"monitor_button_0",
  change:["evt", "color"]}
```

Label interface elements are unique in that they contain text that can be updated. The mMATB Communications Task's channel values provide examples of changing labels. As with the indicator blocks, the ids are known, like "comm_channel_1_frequency" in the example below. However the text of the label changes. In the example below, the change attribute is labeled as event-based (e.g., "evt") because the values rarely change, and only the text of the label is marked for change with "stringval".

```
{type:"Label",
  id:"comm channel 1 frequency",
  change:["evt","stringVal"]}
```

The most dynamic elements in the mMATB interface are the colored circles in the Tracking Task. Each oval moves continuously along a path using the D3 animation library. The constant motion produces a lot of events; this could generate a lot of network traffic and decrease software performance. Therefore, these elements are specified with the "poll" value for the change attribute. The location ("x" and "y") and "color" change, and so all three values are listed in the change attribute. The final attribute of the example specification given below is clickable; this attribute will be described in the next section.

```
{type:"OvalOutlineFill",
  id:"track_circle_0",
  change:["poll","x","y","color"],
  clickAble:true}
```

Keypress and Mouse Events

To complete the interaction loop, actions taken by the model are transmitted to the task environment. There are three types of interaction currently supported by SIMCog-JS: key press, cursor move, and mouse click. The server sends all interactions to the client; the modeler has full control of how to handle (or ignore) events.

The simplest of the three interactions is key press. Key press interactions are handled automatically by the system. This is done by mapping ACT-R keycodes to JavaScript keycodes and dispatching a keydown event to the task. Currently only keydown events are supported; the modeler may modify the client code to support keyup and keypress events.

When a click is performed, a message is sent to the client containing the location of the mouse and the event type (mouseClick). While mouse coordinates may be enough for many tasks, more information is provided, for example, to deal with the asynchronous nature of the system or facilitate a deeper analysis. An example from the mMATB task is when the model clicks on circles in the tracking task that are moving quickly; the circle could move a couple of pixels out from under the cursor before the click event reaches the client. To handle such circumstances, objects can be declared as clickable. Anytime a click is performed by the model, the server determines if the click was performed within any of the clickable objects. If it is determined that one or more objects were clicked, the message to the client will also include the unique IDs of the items clicked, along with the location, type, and ID of every clickable object. This information allows for cases where the unique identifier is needed to click an object within the task and even more complex cases where specific information and computation is desired.

To declare a visual chunk as clickable, add the clickable attribute to an object's specification and set it to true.

```
{..., clickable:true}
```

The client automatically handles clicks by dispatching a JavaScript mouse click event. If a clickable object was clicked, the client dispatches a click event for that element. Otherwise, the client finds the element at the location of the click and simulates the click there.

For mouse movements, JavaScript does not allow control of the cursor in web browsers. Such control is not allowed by code in web browsers for security and usability reasons. To simulate a model's mouse movements in the task, SIMCog-JS generates mouse movement messages for the client. This approach offers both reliability and speed without introducing external software systems.

When the model moves its simulated mouse, a mouseMove message is sent to the client that contains the location of the model's simulated cursor. With this information, the modeler can record the simulated mouse movements similarly to how human mouse movement data are recorded. To do so, the modeler will likely need to modify the client code. For example, in mMATB the cursor-recording code looks like:

```
ws.onmessage = function (evt) {
    //Called when server message received
    var serverMessage = JSON.parse(evt.data);
...
  else if(serverMessage.Command == "mouseMove"){
    track_chart.mouseLocation(
      {x:modelInteraction.mouseX,
      y:modelInteraction.mouseY});
    }}
```

Sending ACT-R Commands

SIMCog-JS supports sending model commands from the task to the model. Doing so is straightforward and takes advantage of existing Java ACT-R methods for executing ACT-R commands. The modeler adds commands to a list in the client code that is sent to the server at the start of execution. For example, to represent the Resource Management Task instructions to maintain the resource level within a target range, the modeler may specify:

```
["(add-dm (resourceTask isa goal
      minLevel 2000 maxLevel 3000))",
 "(goal-focus resourceTask)"]
```

Conclusion and Future Work

SIMCog-JS is a system that allows cognitive models to interact with external software, minimizing the task re-implementation burden on the modeler. The system currently facilitates communication between Java ACT-R and HTML/JavaScript. In addition to describing the architecture of SIMCog-JS, this paper reported on using SIMCog-JS to (a) specify visual interface elements for use by ACT-R and how those interface specifications can be customized, (b) integrate ACT-R responses into JavaScript software, and (c) execute ACT-R commands from the task interface. The strengths of SIMCog-JS are the easy specification of visual objects and interactions with minimal task-code modifications and the seamless interaction between models and browser-based tasks. The modeler need only specify the identity and shape for visual objects to reach ACT-R.

Development is ongoing to improve and extend the functionality of SIMCog-JS. A mid-term goal is to add synchronous execution modes, where the task and model use the same simulation clock, relaxing design requirement 3 without negatively impacting real-time execution. Additional planned features include audio event specification and support for multiple cognitive modeling formalisms, like EPIC architecture (Kieras & Meyer, 1997) and Python-based mathematical models.

By harnessing standard programming protocols and languages, the SIMCog approach can lighten the modeler's burden while broadening the environments in which computational cognitive models operate. Because SIMCog-JS can operate in an environment with facilities for complex data visualization (e.g., D3), we will be pushed to enhance ACT-R's functionality. In the future SIMCog-JS could be integrated with an artificial vision system to, for example, automatically determine object shape; this combined approach could, in fact, bolster both candidate solutions to the task re-implementation challenge.

Acknowledgments

This research was supported AFOSR. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 12/16/2014; 88ABW-2014-5938.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004, October). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Arnegard, R. J., & Comstock, J. R. (1991, May). Multi-attribute task battery: Applications in pilot workload and strategic behavior research. In *6th international symposium on aviation psychology* (pp. 1118–1123). Columbus, Ohio.

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2301–2309.

Büttner, P. (2010). "Hello Java!" Linking ACT-R 6 with a Java simulation. In D. D. Salvucci & G. Gunzelmann (Eds.), *International conference on cognitive modeling* (pp. 289–290). Philadelphia, PA.

Cline, J., Arendt, D. L., Geiselman, E. E., & Blaha, L. M. (2014, May). Web-based implementation of the modified multi-attribute task battery. In *4th annual midwestern cognitive science conference*. Dayton, Ohio.

Fette, I., & Melnikov, A. (2011, November). *The WebSocket Protocol* (Tech. Rep. No. RCF 6455). Internet Engineering Task Force.

Halbrügge, M. (2013). ACT-CV: Bridging the Gap between Cognitive Models and the Outer World. In E. Brandenburg, L. Doria, A. Gross, T. Güntzler, & H. Smieszek (Eds.), *Berliner werk- statt mensch-maschine-systeme* (pp. 205–210). Berlin.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the json network interface. *Behavior Research Methods, 46*, 1007–1012.

JSON-RPC Working Group. (2010, March). *JSON-RPC 2.0 Specification.* http://www.jsonrpc.org/specification/.

Kennedy, W. G., Bugajska, M. D., Adams, W., Schultz, A. C., & Trafton, J. G. (2008). Incorporating Mental Simulation for a More Effective Robotic Teammate. In *Conference on artificial intelligence* (pp. 1300–1305). Chicago, IL.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*(4), 391–438.

Salvucci, D. D. (2013, August). ACT-R: The Java Simulation & Development Environment. *http://cog.cs.drexel.edu/act-r/index.html*.

St Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. L. (2005). Image Processing in Cognitive Models with SegMan. In *Human-computer interaction international* (pp. 1869:1–1869:19).

Veksler, V. D. (2009). Second Life as a Simulation Environment: Rich, high-fidelity world, minus the hassles. In *International conference on cognitive modeling*. Manchester, United Kingdom.

Modeling the Workload Capacity of Visual Multitasking

Leslie M. Blaha (leslie.blaha@us.af.mil) and James Cline (james.cline.ctr@us.af.mil)
711th Human Performance Wing, Air Force Research Laboratory
Wright-Patterson AFB, OH 45433 USA

Tim Halverson (thalverson@gmail.com)
Oregon Research in Cognitive Applications, LLC
Oregon City, OR 97045 USA

Abstract

We utilize the capacity coefficient to characterize the workload capacity of visual multitasking. The capacity coefficient compares cognitive work completed against a baseline parallel model prediction. Capacity coefficient results subsume standard mean response time (RT) dual-task findings while providing a description of workload effects on the whole RT distribution. This yields a theoretically-grounded characterization that can inform computational and process models of multitasking.

Keywords: Workload Capacity; Multitasking; Multi-Attribute Task Battery; Human Information Processing; Dual-Task

Introduction

We seek to provide a better mathematical characterization of cognitive performance during multitasking, the simultaneous execution of more than one decision within the same experimental environment. Often, characterizations of multitasking performance are limited to assessments of dual task decrements, wherein mean response time (RT) or accuracy are compared across only two tasks. An increase (decrease) in mean RT (accuracy) when switching from a single task to the dual-task environment is interpreted as an increase in cognitive workload. This may be further correlated with subjective workload ratings. While these measures do give some indication of participants' experiences of workload, they do not provide strong insight into the cognitive mechanisms supporting the multitasking behaviors or the mechanistic reasons for changes in performance under changing workload demands.

We report on an effort to utilize human information processing modeling to provide qualitative and quantitative characterization of the cognitive mechanisms engaged in multitasking. In particular, we focus on changes in workload capacity, the efficiency with which the system responds to the changing number of tasks in a dynamic environment.

Modified Multi-Attribute Task Battery

To study multitasking, we utilize a web-browser implementation of the modified Multi-Attribute Task Battery (mMATB; Cline, Arendt, Geiselman, & Blaha, 2014), developed in the JavaScript D3 library (Bostock, Ogievetsky, & Heer, 2011). The mMATB consists of four possible visual decision making tasks: Tracking, Monitoring, Communication, Resource Management. In our implementation, all aspects of the workload can be manipulated: entire tasks (quadrants) can be turned on or off, the rate of alerting events can be varied as can the probability of simultaneous alerting events, and the speeds at which the moving parts of the displays move can be adjusted. We will focus herein on manipulations of the total number of tasks to be performed simultaneously.

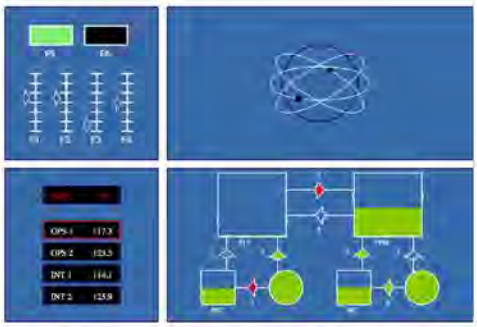

Figure 1: Diagram of the modified Multi-Attribute Task Battery (mMATB) used in the present study. The four visual tasks are (clockwise from upper left): Monitoring, Tracking, Resource Management, Communications.

Figure 1 shows the mMATB environment. The Tracking Task, contained in the upper right, entails physically tracking three colored circles move continuously along individual ellipsoid trajectories. High performance on this task requires continual motion and attention to switching targets.

Both the Monitoring Task (upper left) and the Communications Tasks (lower left) require keypress responses to alert events. In the Monitoring Task, the participant's task is to provide the appropriate response if a parameter is out of its normal state. In the Communications Task, participants must adjust a channel to a new value upon target cuing.

The lower right quadrant contains a Resource Management Task which requires only strategic attention to gates in order to maintain fuel levels within a predetermined range for two schematic resource tanks.

The mMATB, thus, demands a division of visual attention across the four tasks. During multitasking, participants are instructed to emphasize accuracy in the Tracking Task as their primary task, and to respond to all other alerts appropriately. Response times are collected to cued events; response choices are collected for all interactions.

The Capacity Coefficient

Workload capacity is defined as the ability of the cognitive information processing mechanisms to respond to changes in cognitive load. This is usually interpreted as changes in the number of items that need to be processed within a task. Capacity is assessed with RT data, in order to make inferences about information processing speeds. Qualitatively, the possible capacity classes are unlimited, super, and limited capacity, corresponding to processing speeds remaining steady, increasing, or decreasing, respectively.

Our primary measure of workload capacity is the capacity coefficient (Houpt, Blaha, McIntire, Havig, & Townsend, 2014). This is a ratio measure which compares the observed participant's RTs during multitasking to a model-based prediction about multitasking speed. The baseline RT model is an unlimited capacity independent parallel model (UCIP). We utilize the capacity coefficient for ST-ST responses (Blaha, 2010):

$$C(t) = \frac{K_k(t)}{K_{k,C}(t)}. \quad (1)$$

In Equation 1, the numerator gives the cumulative reversed hazard function for individual target channel k when processed alone; this is the UCIP model prediction. The denominator is the cumulative reversed hazard function for target channel k when additional tasks (the set C) are being performed.

Figure 2 illustrates $C(t)$ results for one typical participant in both dual-task multitasking (upper plot) and four-task multitasking (lower plot). Relative to the UCIP baseline at $C(t) = 1$, the data indicate that while while all conditions showed mean RT dual-task decrements, the functional data are more nuanced. Under dual-task conditions, performance in the tracking task improved, showing super capacity $C(t) > 1$ for most times, but falling to limited capacity $C(t) < 1$ when the number of tasks increased to four. Thus, additional task demands have the potential to improve tracking performance.

Detection performance in the monitoring task, on the other hand, was limited capacity in both the dual task and four-task multitasking conditions. Communication task detection was also limited capacity in the four-task condition. This indicates that division of attention across multiple tasks slows alert detection responses.

Discussion

The present work is the first to apply the capacity coefficient to a multitasking situation, where the number of tasks is manipulated while the features within each task (when present) remain unchanged. Current results indicate that some tasks benefit from additional workload demands, while others are slowed. The capacity coefficient can capture both types of effects. This more nuanced characterization can then be used to inform computational and process models, such as the threaded cognition model of multitasking (Salvucci & Taatgen, 2008), and to study task switching and divided attention strategies.

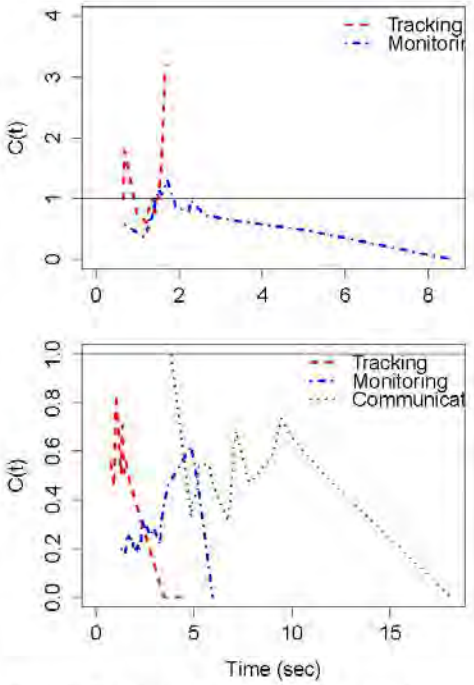

Figure 2: Capacity coefficient results for a typical participant in the dual task (upper) and multitask (lower) conditions.

Acknowledgments

This research was funded by the Air Force Office of Scientific Research. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 12/16/2014; 88ABW-2014-5937.

References

Blaha, L. M. (2010). *A dynamic hebbian-style model of configural learning* (Unpublished doctoral dissertation). Indiana University, Bloomington, Indiana.

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2301–2309.

Cline, J., Arendt, D. L., Geiselman, E. E., & Blaha, L. M. (2014, May). Web-based implementation of the modified multi-attribute task battery. In *Fourth annual midwestern cognitive science conference*. Dayton, Ohio.

Houpt, J. W., Blaha, L. M., McIntire, J. P., Havig, P. R., & Townsend, J. T. (2014). Systems Factorial Technology with R. *Behavior Research Methods*, *46*, 307–330. doi: 10.3758/s13248-013-0377-3

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, *115*(1), 101.

ACT-R and LBA Model Mimicry Reveals Similarity Across Modeling Formalisms

Christopher R. Fisher (christopher.fisher.27.ctr@us.af.mil)
Matthew M. Walsh (matthew.walsh.15.ctr@us.af.mil)
Leslie M. Blaha (leslie.blaha@us.af.mil)
Glenn Gunzelmann (glenn.gunzelmann@us.af.mil)
Air Force Research Laboratory, 711th Human Performance Wing
Wright-Patterson Air Force Base, OH 45433

Abstract

Adaptive Control of Thought-Rational (ACT-R) and the Linear Ballistic Accumulator (LBA) were compared in a model mimicry simulation of the Psychomotor Vigilance Task (PVT), a simple, reaction time (RT) task requiring sustained attention. The models use different formalisms to capture the full response profile of the PVT. The parameters were varied systematically to illustrate the ranges of the models' predictions, to assess the models' estimation properties, and to determine which parameters in the models correspond with each other. Both models produced skewed RT distributions typical of empirical data, including false starts and lapses. The simulation study demonstrated that both models and their parameters are recoverable. Lastly, isolated parameters in the LBA model captured the effects of varying parameters in the ACT-R model, but the reverse was not always true. These interesting correspondences across different modeling formalisms suggest the possibility of integrating ACT-R and the LBA in future work.

Keywords: ACT-R, LBA, PVT, reaction time, fatigue, model comparison

Introduction

The ability to detect a single stimulus is fundamental to cognition. Although this skill is basic, the study and modeling of stimulus detection is worthwhile for several reasons. Stimulus detection has been extensively examined in laboratory tasks involving vigilance and simple reaction time (RT; Luce, 1986). Additionally, this ability underlies successful performance in applied contexts that require sustained attention, such as driving. Finally, intuition suggests that the cognitive processes involved in stimulus detection should be involved in more-complex multi-alternative choices as well.

Despite the simplicity of detection tasks, the RT distributions they produce are complex and empirically rich. This is well-illustrated by the psychomotor vigilance task (PVT; Dinges & Powell, 1985), a 10-minute detection task in which stimuli are presented at random inter-trial intervals ranging from 2 to 10 seconds. Participants are instructed to respond as quickly as possible once the stimulus appears while avoiding premature responses. The PVT response profile consists of three categories: *false starts* occur before or within 150 ms of stimulus presentation, *alert responses* occur between 150 and 500 ms of the stimulus onset, and *lapses* occur 500 ms after of the stimulus onset. The RT distribution on the PVT, which has a long right tail even when participants are well rested, becomes increasingly skewed to the right with greater fatigue from sleep loss, as reflected in increased lapses (Lim & Dinges, 2008). Additionally, participants commit more false starts. These features of the response profile reflect stable individual differences, both at baseline and following sleep loss (Van Dongen, Baynard, Maislin, & Dinges, 2004).

A complete model of the PVT should explain the full response profile, yet most biomathematical accounts from the sleep research literature only predict aggregate measures of performance such as the proportion of lapses (for a review, see Van Dongen, 2004). More recent work has attempted to use statistical functions to characterize the full RT distribution (Lim & Dinges, 2008), but those efforts still fail to explain why the particular distributions arise. A promising alternative is to use computational cognitive models, which specify the cognitive processes underlying task performance, to simulate behavior in the PVT (e.g., Gunzelmann, Veksler, Walsh, & Gluck, 2015).

In this paper, we compared two PVT models derived from very different formalisms. The first model is based on the integrated-cognitive architecture Adaptive Control of Though-Rational (ACT-R), in which RTs are determined by the durations of a sequence of discrete cognitive events. The second model is based on the Linear Ballistic Accumulator (LBA; Brown & Heathcote, 2008), an analytically tractable member of the class of sequential sampling models. In the LBA, RTs are determined by the combined durations of a decision process in which evidence accumulates continuously, and an overall non-decision time attributed to perceptual and motor processes.

The PVT is an ideal test bed for comparing ACT-R and the LBA because (1) the PVT is simple, yet (2) it provides empirically rich data for inferring cognitive processes, and (3) both ACT-R and the LBA can be applied to the PVT. Rather than attempting to falsify one account, we sought to compare and contrast these differing formalisms.

We addressed three primary questions in this research. First, can both ACT-R and LBA generate the complete RT profiles, including false starts and lapses, observed in PVT studies? ACT-R models have predominantly been used to predict mean RTs, and attempts to account for full RT distributions have been rare (but see Walsh et al., 2014). The LBA has only been used to model the correct and error responses in multi-alternative choice tasks (Brown & Heathcote, 2008), and it was unclear whether it could also account for the full response profile observed in the PVT, especially the occurrence of false starts and lapses. Second, how well can ACT-R and LBA recover their own parameters from simulated PVT data? Both models are complex, and the estimation properties of their parameters have not been assessed in the PVT. As such, it was unknown whether model parameters could be reliably estimated from PVT data, or whether the models could even be distinguished from one another based on data from the PVT. Third, what are the relationships between core parameters in the two models? Although the models are distinct, it was unclear which of their parameters are conceptually and/or functionally linked.

Models

LBA

The LBA is a sequential sampling model that is similar to the drift diffusion model (DDM) in terms of parameter interpretation (Brown & Heathcote, 2008; Donkin et al., 2011). In both models, information is sampled from a stimulus and accumulates over time. When accumulated evidence in favor of an alternative reaches a threshold, a decision occurs. Sources of variation in the DDM, such as intra-trial variability in evidence accumulation and inter-trial variability in non-decision time, are absent from the LBA. These simplifications come with no loss of generality, making LBA a more parsimonious, complete account of basic empirical RT phenomena (Brown & Heathcote, 2008).

In the standard LBA, the stimulus onset triggers an evidence accumulation process. Accumulated evidence begins from a variable starting point between 0 and the response threshold, and proceeds towards the response threshold in a linear and deterministic fashion. The speed of the accumulation process is controlled by the drift rate. Between-trial variability in the drift rate and starting point of the evidence accumulation process contribute to the shape and spread of the RT distribution. The drift rate is normally distributed across trials with a mean of V, and a standard deviation of 1. The starting point is uniformly distributed with an adjustable maximum starting point, A. Other processes such as encoding and motor execution are combined into a composite measure of non-decision time, t_0.

Several modifications were necessary to apply the LBA to the PVT (Fig. 1). Our modified LBA model involves two accumulation processes that occur in succession rather than one accumulation process. First, an inter-stimulus interval (ISI) accumulation process starts at the beginning of the trial. Although this process has a negative drift rate on average, stochasticity occasionally results in a positive drift rate and, consequently, a false start. Once the stimulus appears, the ISI accumulation process halts and a separate stimulus interval (SI) accumulation process starts. The trial ends once a response is given.

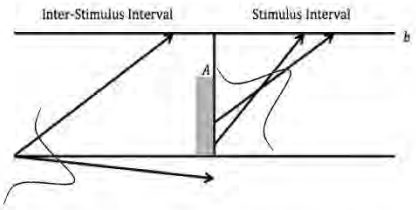

Figure 1. The modified LBA has separate accumulators for the inter-stimulus and stimulus intervals. A denotes spread of start points for stimulus interval, and b denotes threshold for both intervals. The vertical bar marks stimulus onset.

The ISI and SI accumulation processes are identical, except for mean drift rate, V, and the maximum starting point, A. The ISI mean drift rate, V_{ISI}, is constrained to be negative, indicating that false starts are rare and produced randomly. Additionally, the ISI maximum starting point, A_{ISI}, is set to zero to reflect bias toward not responding. The threshold, b, is the same for the ISI and SI accumulation processes, as is non-decision time, t_0. In total, the modified LBA model contains five free parameters: b, A_{SI}, V_{ISI}, V_{SI}, and t_0.

ACT-R

ACT-R contains a set of specialized information-processing modules (e.g., a vision module, a declarative memory module, a motor module). These modules are connected to, and controlled by, a central procedural module (Anderson, 2007). Procedural knowledge is represented in the form of production rules, which consist of selection criteria and actions that modify the internal state of the architecture and the external state of the world when the selection criteria are met. The temporal dynamics of cognition unfold across a sequence of production cycles. During each cycle, the conditions for each production are compared against the conditions of the current state, and a production is selected and enacted if its conditions are met. The resulting state serves as the starting point for the next production cycle.

We adopted an ACT-R model of the PVT that consists of three productions: (1) wait for the stimulus to appear, which represents task engagement, (2) attend to the stimulus, and (3) respond to the stimulus (Walsh et al., 2014). Partial production matching allows productions whose conditions are not perfectly met to be selected in a stochastic fashion, producing occasional false starts. The probability that a production is selected is modulated by two adjustable parameters—a utility scalar (U_S) and a utility threshold (U_T). Formally, production utility can be expressed as:

(1) $U_{ij} = U_S(U_i - MMP_{ij}) + \epsilon_i$

where U_{ij} is the utility of production i in state j, U_S is the utility scalar, U_i is the stored utility for production i, MMP_{ij} is the mismatch penalty for production i in state j, and ϵ_i is logistically distributed noise. The resulting payoff matrix is symmetric with 0 assigned to mismatches and 1 assigned to matches. The mismatch penalty ensures that productions whose conditions are not perfectly met will be selected with low probability.

The production with highest utility is selected and enacted if its utility exceeds the utility threshold, U_T,

(2) $\text{Production} = \max(U_{ij}) \text{ if } \max(U_{ij}) > U_T$

If no production's utility exceeds the utility threshold, a microlapse occurs and no production is enacted. Following a microlapse, the utility scalar in Eq. 1 is decremented by an adjustable scalar, FP_{dec}, according to $U_S - U_S \cdot FP_{dec}$. This increases the likelihood of microlapses in subsequent production cycles. Across such a series of cycles, the probability of responding decreases progressively, causing behavioral lapses. The final adjustable parameter, *cycle time*, controls the duration of conflict resolution at the start of each production cycle. In total, the ACT-R model contains four free parameters: U_S, U_T, FP_{dec}, and *cycle time*.

Our model harnessed two sources of temporal variability. The first related to the variable sequence of productions selected in a trial, and the second related to the stochastic duration of production and cycle times. Each trial's RT, then, was determined by the summed durations of the productions and their associated cognitive and motor processes. In this way, the ACT-R model can produce a full distribution of RTs, rather than an approximation of an aggregate mean RT (Walsh, et al., 2014).

Simulation Method

We simulated an idealized selective influence experiment (Donkin, et al., 2011) in which the parameters of each model were systematically varied one at a time while all others were set to default values. This approach allowed us to examine (1) our ability to accurately recover parameters of each model, (2) the extent to which the models mimicked each other and (3) how the parameters were correlated between models. Parameter ranges were drawn from the published model fits of PVT performance by 13 well-rested individuals in the control condition of a sleep deprivation experiment (Doran, Van Dongen, & Dinges, 2001; see also Walsh et al., 2014). We set the default value of each parameter to the median estimate from the individual model fits, and the range of each parameter to the complete range of estimates from the individual fits (Table 1). We varied parameters at ten equally spaced intervals over their ranges, resulting in 40 ACT-R parameter sets (10 levels per parameter by 4 parameters) and 50 LBA parameter sets (10 levels per parameter by 5 parameters). We simulated 50,000 PVT trials for each model and parameter set to minimize the role of sampling error and bias in our analyses.

Table 1. Default parameters and ranges in the simulation.

LBA	b	A_{SI}	V_{SI}	t_0	V_{ISI}
Default	0.68	0.44	3.42	0.15	-2.34
Min	0.54	0.1	3	0.15	-2.95
Max	0.98	0.56	3.9	0.18	-2.01
ACT-R	U_S	U_T	FP_{dec}	***Cycle Time***	U_S - U_T
Default	4.85	4.39	0.98	0.04	0.46
Min	4.01	4.07	0.91	0.029	-0.38
Max	5.6	5.02	0.99	0.057	1.21

Each model was fit to the 90 simulated datasets using quantile maximum likelihood estimation (Heathcote, Brown & Mewhort, 2002). RTs that occurred prior to stimulus onset or within 150 ms of stimulus onset were combined into a false start bin (Lim & Dinges, 2008). The remaining portion of the distribution was further divided into 20 quantile bins. Likelihood estimates were calculated from the observed and expected proportions of RTs within each quantile bin. A simplex algorithm embedded within a grid search was used to find the model parameters that maximized the likelihood of each simulated dataset. Large-scale computing resources (Harris, 2008) were leveraged for ACT-R, as it is computationally intensive.

Results

Model RT Distributions

Figure 2 shows four of the most distinctive RT distributions produced by ACT-R and the LBA. The distributions, which vary in terms of numbers of false starts and lapses as well as median RTs (Table 2), are within the ranges of those produced by well-rested and sleep deprived individuals (cf., Walsh et al., 2014). In the 90 simulated datasets, the models produced similar proportions of false starts and lapses and similar median RTs. However, the LBA model consistently yielded distributions with more pronounced skew.

Table 2. Proportions of false starts and lapses, and median RTs from the simulated distributions in Fig. 2.

Model	**Curve**	**False Starts**	**Lapses**	**Median RT (ms)**
ACT-R	Blue	.006	.000	245
	Red	.008	.005	272
	Black	.010	.083	305
	Green	.101	.222	381
LBA	Blue	.006	.000	242
	Red	.008	.010	271
	Black	.011	.085	306
	Green	.106	.210	381

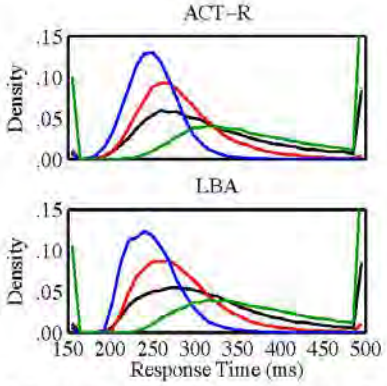

Figure 2. Proportion of RTs in 10 ms bins ranging from 150 ms to 500 ms. The first bin contains all RTs before 150 ms, and the last bin contains all RTs after 500 ms. Blue, red, black, and green lines show fast, medium, slow and sleep deprived RT distributions.

Parameter Recovery

The parameter recovery model fits address how accurately the parameters can be estimated from PVT data. In these analyses, the models were fit to their self-generated data. Two metrics were used to assess the quality of the parameter recovery: correlation to measure the linear association between the true and recovered parameters, and relative bias to measure the precision of the estimates.

Table 3 (upper) shows the parameter recovery results for ACT-R. The high correlation for *cycle time* indicates that this parameter is recoverable. Correlations for U_S and U_T were moderate, but the correlation for the difference between U_S and U_T was high. This indicates that the utility scalar and threshold jointly influence performance dynamics in the ACT-R model. The low correlation for FP_{dec} is due to the relatively infrequent occurrence of lapses in well-rested individuals. Relative bias was low across all parameters, indicating the high precision of the estimates.

Table 3 (lower) displays the parameter recovery results for the LBA. The high correlations and low relative bias indicate that the parameter recovery was successful. Collectively, these results show that parameters from both models can be reliability estimated from their own simulations of PVT data.

Table 3. Parameter recovery results for ACT-R and LBA.

ACT-R	U_S	U_T	FP_{dec}	*Cycle Time*	U_S - U_T
Correlation	0.85	0.77	0.56	0.99	0.99
Relative Bias	1%	1%	0%	0%	4%
LBA	b	A_{SI}	V_{SI}	t_0	V_{ISI}
Correlation	0.93	0.97	0.85	0.85	0.98
Relative Bias	-3%	2%	-1%	3%	0%

Model Mimicry

The model mimicry analyses address whether ACT-R and the LBA produce different predictions on the PVT. In these simulations, the ACT-R and LBA models were cross-fit to data generated by each other. The Bayesian Information Criterion (BIC) was used to determine whether the data-generating model provided a better fit to the RT distributions than the alternate model while adjusting for parametric sources of model complexity. Smaller values denote better fit.

Figure 3 shows the BICs averaged across datasets for each model. In all 90 simulated data sets, both models provided better fits to their own data than the alternate model. This shows that although the models make very similar predictions they are identifiable in simulations with very large sample sizes.

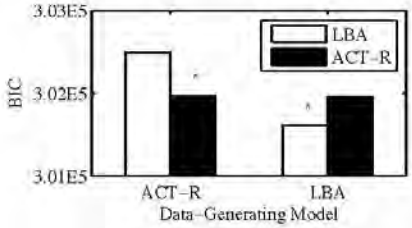

Figure 3. BIC averaged across datasets. Stars denote fit of data-generating model to itself.

Parameter Correspondence

We examined the manner in which parameters in the two models corresponded to one another. In our simulations, parameters were varied one at a time while the other parameters were fixed. In the simplest case, a change in one parameter would be captured by variation in a single, analogous parameter in the alternate model. For simplicity, we considered three core parameters in the ACT-R model (U_S - U_T, FP_{dec}, and *cycle time*), and four in the LBA (V_{SI}, V_{ISI}, t_0, and $b - A_{SI}/2$). The composite parameter $b - A_{SI}/2$, called *response caution*, is derived from the threshold and the center of the start point distribution, and measures the average amount of information that is needed to reach the decision threshold (Donkin, et al., 2009).

We first examined how ACT-R responded to manipulations of the LBA parameters (Table 4). No

Table 4. Correlations between LBA (data generating) and ACT-R (best fitting) parameter values. *p <.05

	ACT-R		
LBA	FP_{dec}	*Cycle Time*	U_S - U_T
V_{ISI}	-0.06	0.08	0.04
V_{SI}	0.10	-0.09	0.16
t_0	0.04	0.20	0.22
Response Caution	-0.63*	0.91*	0.68*

Table 5. Correlations between ACT-R (data generating) and LBA (best fitting) parameter values. *p <.05

ACT-R	LBA			
	V_{ISI}	V_{SI}	t_0	*Response Caution*
FP_{dec}	-0.22	0.16	0.08	0.30
Cycle Time	-0.18	-0.01	-0.08	0.89*
U_S - U_T	0.07	0.96*	-0.41*	0.08

parameters in the ACT-R model were selectively influenced by changes to V_{SI},V_{ISI} and t_0, but all parameters were affected by changes to *response caution*. Next, we examined how the LBA responded to manipulations of ACT-R parameters (Table 5). Changes to *cycle time* were captured by *response caution*, and changes to U_S - U_T were captured by V_{SI}. No parameter in the LBA was selectively influenced by changes to FP_{dec}. In sum, there was a direct mapping between individual ACT-R parameter manipulations and LBA parameters, but not between individual LBA parameter manipulations and ACT-R parameters.

Discussion

The detection of a single stimulus is among the most-widely studied topics in cognitive science. Yet, despite the simplicity of one-choice RT tasks, the RT distributions they produce are complex and difficult to account for in detail. Here, we compared two computational cognitive models of the PVT. One model was based on ACT-R and consists of a sequence of discrete cognitive events while the other was based on the LBA, which involves continuous evidence accumulation. The results of our simulations support three findings. First, both models produced the qualitative shapes of RT distributions found in the PVT, including the long right tail of RT distribution, and occasional false starts and lapses (Fig. 2). Second, most model parameters were recoverable and the PVT was capable of distinguishing between the models. Third, isolated parameters in the LBA model captured the effects of varying ACT-R parameters, but the reverse was not always true. The correspondence between ACT-R parameters and LBA parameters suggests similarity between these differing modeling formalisms.

Model Comparison

The correspondence between parameters in the LBA and ACT-R models was complex. In some cases, parameters in one model were affected by parametric variations in the other in intuitive ways. For example, drift rate (V_{SI}) in the LBA captured changes in the difference between the utility scalar and threshold (U_S - U_T) in ACT-R. This makes sense because both fundamentally control the signal-to-noise ratio in the decision process.

In other cases, unexpected model parameters corresponded to one another. For example, changes in *response caution* in the LBA were captured by *cycle time* in ACT-R and vice versa. *Response caution* is thought to be sensitive to instructions designed to prioritize speed or accuracy, whereas *cycle time* is conceptualized as a stable property of the cognitive architecture that only varies among individuals. ACT-R posits that production selection is instantiated in the basal ganglia, which receives input from multiple excitatory and inhibitory pathways. It is conceivable that the duration of production selection, represented by *cycle time*, varies with dynamic activity from these pathways. In other words, the relationship between *response caution* and *cycle time* may be real, despite the current standard of fixing *cycle time* within ACT-R models of individuals.

In a third set of cases, we found little correspondence between model parameters. For example, ACT-R failed to capture manipulations of non-decision time in the LBA. This relationship was relatively symmetrical in that non-decision time showed little or no systematic relationship to the manipulation of any ACT-R parameters. Such a lack of correspondence suggests that an experimental manipulation of non-decision time could potentially discriminate between ACT-R and the LBA. Moreover, this finding indicates that conclusions will depend critically upon which model is used to evaluate data.

Effects of Fatigue on Psychomotor Vigilance

We demonstrated that the ACT-R and LBA models produce a range of response profiles that are similar to each other, and similar to those observed in well-rested individuals. The models rarely responded before 150 ms of stimulus presentation (false starts), and they rarely responded more than 500 ms after the stimulus appeared (lapses). False starts and lapses, though present in baseline RT distributions, are greatly exacerbated by fatigue from sleep loss. As shown by Walsh et al. (2014), ACT-R can be integrated with a biomathematical model of fatigue to predict the effects of time awake and time of day on PVT performance. The LBA model has not been expanded to account for the effects of fatigue on PVT performance, yet it should be conceptually straightforward to do so.

Evaluating the models under conditions of fatigue might also enhance model discriminability. More confidence can be placed in a model that captures normal as well as impaired cognitive functioning. Certain parameters that are essential to capturing the effects of fatigue minimally affect alert performance on the PVT (FP_{dec} and U_T in ACT-R, and V_{ISI} in the LBA). In this sense, sleep deprivation protocols provide a unique opportunity to distinguish among models of the PVT (Walsh et al., 2014) and could be leveraged as a general strategy for model comparison.

Towards an Integration of ACT-R and the LBA

Sequential sampling models and ACT-R explain cognition using different modeling formalisms. Sequential sampling models provide detailed accounts of empirical RT distributions. This emphasis comes at the cost of limited generalizability beyond well-constrained decision-making tasks utilizing fixed trial structures. Cognitive architectures,

by contrast, focus on the unification and generalization necessary to model complex tasks. Because of this focus, cognitive architectures neglect certain details of low-level decision processes.

Efforts to capitalize on the complimentary strengths of sequential sampling models and cognitive architectures have been made recently. Van Maanen, van Rijn, and Taatgen (2012) combined the DDM and ACT-R to form RACE/A, which accounts for the dynamics of declarative memory in a picture-word interference task. A DDM with multiple accumulators governs how the activation values of information in declarative memory change over time and determine retrieval latencies. ACT-R, in turn, provides the control structure necessary for coordinating the multitude of decision and non-decision processes evoked by the task.

Within the context of the PVT, sequential sampling models could be used as a mechanism for production selection. Presently, the duration of production selection in ACT-R is treated as a uniform random variable with a mean of about 40 ms (Table 1). Each production could instead be represented as an accumulator with a drift rate determined by the match between the state of the world and the production's conditions. Integrating these approaches would provide a theory of production selection (implemented as a sequential sampling model) along with a theory of task control (implemented as production rules). The LBA would be a natural choice for the sequential sampling model for three reasons: (1) it is applicable to selection among two or more alternatives, (2) it is more parsimonious than other sequential sampling models, and (3) parameter estimation is efficient and mathematically tractable.

Incorporating a sequential sampling model into a cognitive architecture would provide a more detailed, formal account of the time course of production selection. Such an account would provide a rationale for changes in the stochastic duration of cycle time. Although such an account may be unnecessary for modeling the PVT, incorporating both representational levels would be useful for capturing complete performance dynamics in more complex tasks. Factors in multi-alternative choice tasks such as decision conflict and value influence decision times (Ratcliff & Frank, 2012). Likewise, factors in single-alternative choice tasks such as stimulus contrast and luminosity influence decision times. Presently, these effects are difficult to explain in ACT-R. Implementing production selection as a sequential sampling process could overcome these challenges.

Acknowledgments

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. M.M.W. held a National Research Council Research Associateship Award with the AFRL while conducting this research. This research was supported by an AFOSR grant to L.M.B. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 03/09/2015; 88ABW-2015-0914.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* NY, NY: Oxford University Press.

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive Psychology, 57*, 153-178.

Dinges, D. F., & Powell, J. W. (1985). Microcomputer analyses of performance on a portable, simple visual RT task during sustained operations. *Behavior Research Methods, Instruments, & Computers, 17*, 652-655.

Donkin, C., Brown, S., Heathcote, A., & Wagenmakers, E.-J. (2011). Diffusion versus linear ballistic accumulation: different models but the same conclusions about psychological processes? *Psychonomic Bulletin & Review, 18*(1), 61-69.

Gunzelmann, G., Veksler, B. Z., Walsh, M. M., & Gluck, K. A. (*2015*). Understanding and predicting the cognitive effects of sleep loss through simulation. *Translational Issues in Psychological Science, 1*(1), 106-115.

Harris, J. (2008). MindModeling@Home: A large-scale computational cognitive modeling infrastructure. In *Proceedings of the Sixth Annual Conference on Systems Engineering Research 2008* (pp. 246–252). Los Angeles, CA, USA.

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2002). Quantile maximum likelihood estimation of response time distributions. *Psychonomic Bulletin & Review, 9*, 349-401.

Lim, J., & Dinges, D. F. (2008). Sleep deprivation and vigilant attention. *Annals of the New York Academy of Sciences, 1129*, 305-322.

Luce, R. D. (1986). *Response times*. New York, NY: Oxford University Press.

Ratcliff, R., & Frank, M. (2012). Reinforcement-based decision making in corticostriatal circuits: Mutual constraints by neurocomputational and diffusion models. *Neural Computation, 24*, 1186-1229.

Ratcliff, R., & Van Dongen, H. P. A. (2011). Diffusion model for one-choice reaction-time tasks and the cognitive effects of sleep deprivation. *Proceedings of the National Academy of Sciences, 108*, 11285-11290.

Van Dongen, H. P. A., Baynard, M. D., Maislin, G., & Dinges, D. F. (2004). Systematic interindividual differences in neurobehavioral impairment from sleep loss: Evidence of trait-like differential vulnerability. *Sleep, 27*, 423-433.

Van Maanen, L., van Rijn, H., & Taatgen, N. (2012). RACE/A: An architectural account of the interactions between learning, task control, and retrieval dynamics. *Cognitive Science, 36*, 62-101.

Walsh, M. M., Gunzelmann, G., & Van Dongen, H. P. A. (2014). Comparing accounts of psychomotor vigilance impairment due to sleep loss. In *36th Annual Conference of the Cognitive Science Society*, Quebec City, Canada.

Exploring Individual Differences via Clustering on Capacity Coefficients

Joseph W. Houpt (joseph.houpt@wright.edu)
Wright State University, Department of Psychology, 3649 Colonel Glenn Highway, Dayton, OH 45435 USA

Leslie M. Blaha (leslie.blaha@us.af.mil)
711th Human Performance Wing, Air Force Research Laboratory, 2255 H Street, Wright-Patterson AFB, OH 45433 USA

Abstract

The capacity coefficient is a well-established, model-based measure that compares performance with multiple sources of information together to performance on each of those information sources in isolation. The measure is a function across time and can potentially carry a large amount of information about a participant. In many applications, this information has been ignored, either by using qualitative assessment of the function or by using a single summary statistic. Recent work has demonstrated the efficacy of dimensional reduction, particularly functional principal components analysis, for extracting important information about the capacity function. We extend this work by applying additional techniques from statistical learning, including K-means and hierarchical clustering to examine individual differences in configural learning.

Keywords: Configural Learning; Individual Differences; Capacity Coefficient; Human Information Processing Modeling

Introduction

The basic goal of the capacity coefficient is to compare response times (RTs) with multiple sources of information to RTs with a single isolated source of information (Townsend & Nozawa, 1995; Townsend & Wenger, 2004; Houpt & Townsend, 2012; Houpt, Blaha, McIntire, Havig, & Townsend, 2013). There are a number of factors that can change performance with increased workload, including factors such as correlated processing of the sources information, processing strategy, and task demands. An additional factor, which is of less interest to the study of cognition, is the effect of statistical facilitation/inhibition. This is the basic phenomenon that the fastest (slowest) sample of multiple random processes tends to be faster (slower) than a sample from any of the individual random processes. The capacity coefficient controls for speed up or slow down from statistical facilitation or inhibition by measuring performance relative to the predicted performance of an unlimited capacity, independent parallel (UCIP) model.

More formally, if the time to process A, T_A, has the cumulative distribution $F_A(t) = P(T_A \leq t)$ and likewise for B, and A and B are independent, then the probability that neither has finished is:

$$\begin{aligned} 1 \quad & F_{AB}(t) = P(T_A > t \text{ and } T_B > t) \\ & = P(T_A > t)P(T_B > t) = (1 - F_A(t))(1 - F_B(t)) \end{aligned} \quad (1)$$

The capacity coefficient is a ratio measure comparing the observed RTs with multiple sources to the predicted performance of the UCIP system estimated from the observed response with only a single source (i.e., Equation 1). This comparison is usually in the form of a ratio of the cumulative hazard functions, defined by $H(t) = -\log[1 - F(t)]$.

$$C_{or}(t) = \frac{H_{AB}(t)}{H_A(t) + H_B(t)} \quad (2)$$

If A and B must be exhaustively processed (i.e., both processes are required to finish) we have a different baseline prediction from the UCIP process.

$$\begin{aligned} F_{AB}(t) &= P(T_A \leq t \text{ and } T_B \leq t) \\ &= P(T_A \leq t)P(T_B \leq t) = F_A(t)F_B(t) \end{aligned} \quad (3)$$

Using the cumulative reverse hazard function, $K(t) = \log[F(t)]$, we can state the capacity coefficient for exhaustive tasks in a form similar to Equation 2.

$$C_{and}(t) = \frac{K_A(t) + K_B(t)}{K_{AB}(t)} \quad (4)$$

Another case of interest is when the target information is presented either alone or with either distracting or irrelevant information. In this case, the UCIP prediction is that the RT distribution will be the same regardless of whether or not there are additional, non-target sources of information presented.

$$F_{AX}(t) = P(T_A \leq t) = F_A(t) \quad (5)$$

For these single-target, self-terminating (STST) cases, the capacity coefficient is defined in terms of the cumulative reverse hazard function.

$$C_{stst}(t) = \frac{K_A(t)}{K_{AX}(t)} \quad (6)$$

In early capacity coefficient applications, the analysis was limited to plotting the function and visually assessing the function in comparison to the baseline of 1. More recently, Houpt and Townsend (2012) developed summary test statistics for comparing performance to a null hypothesis of UCIP processing. While this statistic is certainly an improvement over purely visual assessment, it loses much of the information about the shapes of the functions. Even among participants who fall into the "significantly above baseline" category based on the summary test statistic there can a wide variety of functional shapes.

Burns, Houpt, Townsend, and Endres (2013) demonstrated the use of functional principal components analysis (fPCA; Ramsay & Silverman, 2005) for analyzing differences in the forms of the capacity functions. fPCA is similar to standard principal components analysis (PCA) in that it is a rotation of

the basis space used to describe the observed data. The difference is that in PCA the bases are vectors whereas in fPCA the bases are functions. More succinctly, in PCA, the data are described as a linear combination of orthogonal vectors which are ordered by the amount of variance in the data along that vector. In fPCA, the data are described as a linear combination of orthogonal functions ($\int f_i f_j = 0$) which are ordered by the amount of variance in the data along that function (i.e., f_i maximizes $\sum_{k=1}^{N} (\int f_i x_k)^2$ where x_k are the observed functions, subject to the constraint that $\int f_i f_j = 0$ for $j < i$).

fPCA and PCA are often used to describe a dataset with a dimensional subspace than the original data by only using the first n bases (effectively projecting the data onto a lower dimensional subspace). Each individual datum can then be described by its factor scores on those n bases. For example, if $x_i = a_1 f_1 + a_2 f_2 \ldots a_m f_m$ where the f_j are the basis functions from fPCA, then we can use a lower dimensional representation of x_i given by $x_i \approx a_1 f_1 + a_2 f_2$. fPCA reduction can provide us with a tractable vector space together with representative functions to describe capacity coefficient data. In particular, similarity in the vector fPCA score space captures similarity in capacity function shapes, thereby providing a way to quantify properties of the full functions.

Clustering

Our present effort explores the use of two popular clustering methods, K-means clustering and hierarchical clustering applied to the fPCA-reduced capacity coefficients with a goal of systematically and quantifiably capturing patterns of similarity and differences in capacity coefficients only previously described in qualitative ways. K-means clustering refers to a technique in which a set of points (in any finite dimensional vector space) are modeled as belonging to one of K different clusters. The free parameters of the model are the locations of the center of each of the K clusters, chosen to minimize the Euclidean distance between each datum and its nearest cluster center. The number of clusters to use, K, is experimenter-specified, either using a scree plot or comparing the ratio of within cluster variation to between cluster variation across different values of K.

Hierarchical clustering is an alternative to the K-means approach which is based on building successively more inclusive grouping of data (agglomerative) or successively dividing the data into more exclusive groupings (divisive). We use a basic agglomerative procedure which first clusters the closest nodes. The next cluster is formed by either grouping a different pair of nodes which have the next smallest distance between them or by clustering a datum with the previously formed cluster if the distance between the datum and the cluster is less then the distance between any pair of data. This procedure iterates until a single cluster forms.

Configural Learning Data

We analyzed the data from a configural learning study by Blaha (2010) in which workload capacity qualitatively changed over the course of training. Configural learning is the process by which individual object features are "chunked together" or unified into a single perceptual unit. Configural learning through unitization changes the perceptual representation of the objects, and Blaha and colleagues demonstrated that this not only changes the information processing mechanisms supporting object classification (Blaha, 2010) but also changes the supporting scalp-level neural responses (Blaha, Busey, & Townsend, 2009).

The experiment entailed two categorization tasks based on Goldstone (2000). A conjunctive categorization task was designed to require exhaustive processing of the object belonging to category 1 by systematic variation of the category 2 object features. Mandatory exhaustive processing of this object encouraged participants to chunk the features into a single object; thus, we expected (and previously observed) unitization of this object. Unitization results in a reduction of perceptual workload and an increase in processing efficiency over the course of learning, captured by capacity coefficients that shifted from limited to super capacity levels over training.

A single-feature categorization task served as a baseline estimate for learning a single feature within objects similar to the conjunctive task. Each category in this task only contained a single object, with one feature differing between the two objects. Thus, RTs in this task captured the speed of responding as participants learned to distinguish individual visual features. Single-feature task RT distributions were used to formulate the UCIP estimates for the capacity coefficients.

A total of fourteen participants completed 10-14 experimental sessions, including 5-7 training sessions of both the conjunctive and single-target categorization tasks. Each one-hour session consisted of 1200 trials. In all, the statistical learning herein utilized 12,000-16,800 trials for each of the 14 participants (see Blaha, 2010, for full study details).

For every day of training, four capacity coefficients were estimated for each participant. First, based on the mandatory exhaustive stopping rule, the unitized object was examined with $C_{\text{and}}(t)$. The complementary responses (i.e., category 2, non-conjunctive objects) required the identification of features unique to category 2, engaging an STST response rule. Thus, learning in category 2 was analyzed with $C_{\text{stat}}(t)$. For both $C_{\text{and}}(t)$ and $C_{\text{stst}}(t)$, absolute and relative capacity coefficients were estimated. Absolute learning measured changes in the $C_{\text{and}}(t)$ with the UCIP estimate derived from the first training day, to give an overall estimate of capacity improvement from the start of the learning process. Relative learning varied the UCIP estimate, to account for the single-target discrimination learning occurring in parallel with configural learning; relative $C(t)$ values used numerators and denominators from corresponding training days.

Figure 1 illustrates the AND capacity data for all participants. Day 1 of training is shown in the thinnest line, and the last day of training is the thickest in each plot. All participants exhibited $C_{\text{and}}(t)$ improvements over training, but as Figure 1 highlights, there was a variety of individual dif-

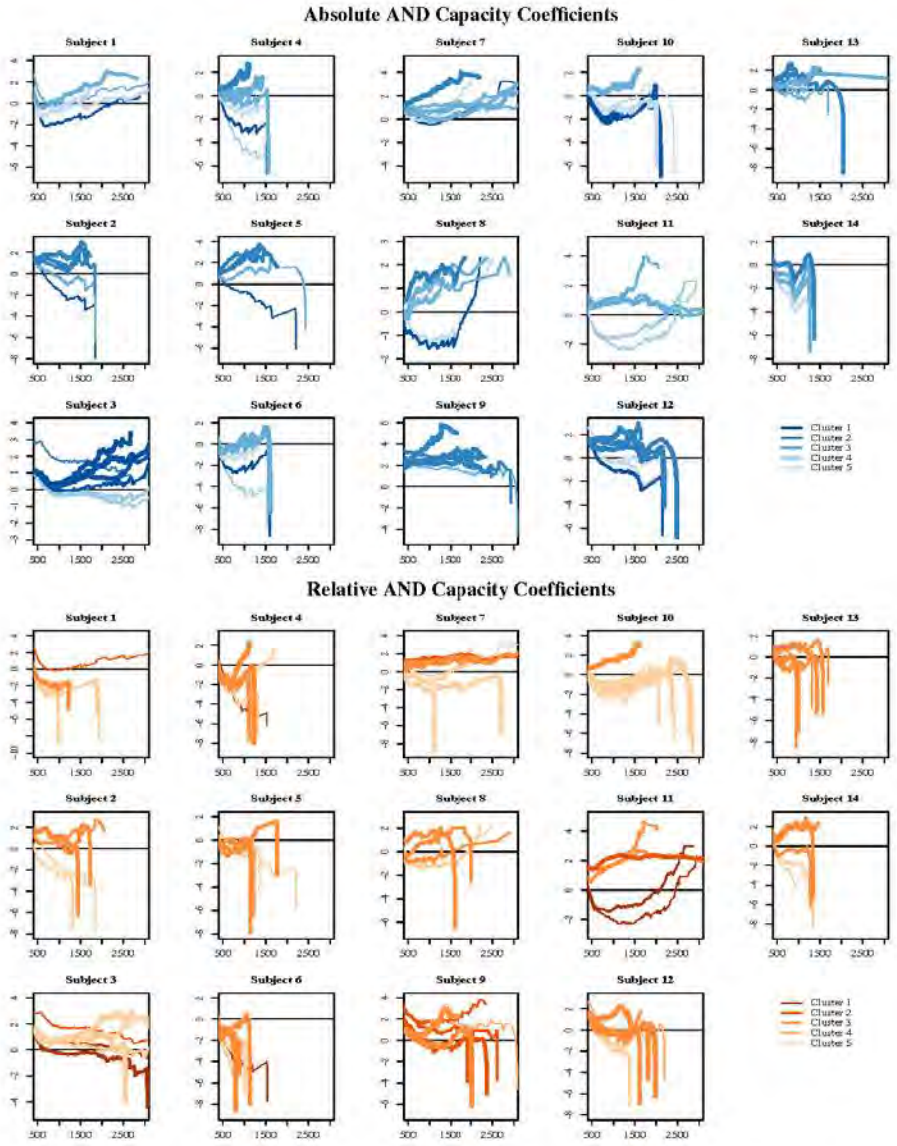

Figure 1: AND Capacity coefficients for all participants over all training. The upper half gives the absolute $C_{\text{and}}(t)$ data, and the lower half gives the relative $C_{\text{and}}(t)$ data. The thickness of each line indicates the training session where the thinnest lines are the first session and the thickest line in each plot is that participant's final day of training. Line colors indicate the $K = 5$ K-means cluster assignment for each $C_{\text{and}}(t)$ curve.

ferences observed by Blaha (2010). For example, Subject 4 exhibited a gradual improvement from limited $C_{\text{and}}(t) < 1$ to

super $C_{and}(t) > 1$, whereas Subjects 8 and 11 showed a more step-like shift from limited directly to super capacity. Subjects 9, 3, and 12 had strong speed-accuracy trade-offs, with super capacity early in training at the cost of lower accuracy.

By applying unsupervised learning to systematically determine the numbers of unique patterns in the data we can quantify these verbal descriptions of the various learning patterns that would otherwise be merely observational inferences.

K-means Clustering

For all four $C(t)$ estimates, we considered the first 4 fPCs, creating four 4-dimensional vector spaces in which we could compare the capacity data. fPCA analysis was performed separately for each of the four types of capacity coefficients, and so we first analyze the unique components within each of the those $C(t)$ classes. K-means analysis was used to identify the number of unique $C(t)$ function shapes exhibited within each condition.

In all conditions, $K = 5$ clusters of functions emerged. Figure 2 illustrates the five clusters for both the absolute (top) and relative (bottom) $C_{and}(t)$ fPCA score spaces, with similar results for $C_{stst}(t)$ fPCA scores. The capacity coefficients plotted in Figure 2 illustrate the $C_{and}(t)$ functions representative of the centroids of each cluster. These were computed by $x_i \approx a_1 f_1 + a_2 f_2 + a_3 f_3 + a_4 f_4$ where $\{a_1, \ldots, a_4\}$ are the 4D centroid score values.

The shapes of the centroid capacity functions (Figure 2) are consistent with the generally observed trends over learning. One cluster shows strict limited capacity $C_{and}(t) < 1$ values, consistent with the inefficient performance early in training. Other clusters show mixed values above and below 1, reflecting the functions in the middle of training that tend to shift from limited to unlimited to super capacity, as well as often showing non-flat shapes (e.g., super capacity for fast RTs, limited capacity for slow RTs). A final cluster exhibits strict super capacity $C_{and}(t) > 1$ values, consistent with participants reaching highly efficient processing by the end of training. K-means clustering shows that the raw capacity coefficient data in configural learning can be classified into 5 fundamental shapes.

Hierarchical Clustering

For the hierarchical analysis, we mapped the functional learning traces into a high-dimensional linear space by aggregating each participant's fPCA scores for each capacity coefficient over all days of training. Participants exhibiting similar functional learning traces are represented by vectors close in this space. Note that because fPCA scores further represent a standardization of $C(t)$ functions, we can map both the $C_{and}(t)$ and $C_{stst}(t)$ learning into the same high-dimensional space.

Hierarchical clustering was performed on 20-dimensional fPCA score space. In this space, each participant was represented by four vectors, one for each type of capacity coefficient (relative and absolute $C_{and}(t)$ and $C_{stst}(t)$). The 20D vectors contained the four fPC weights over five days of training

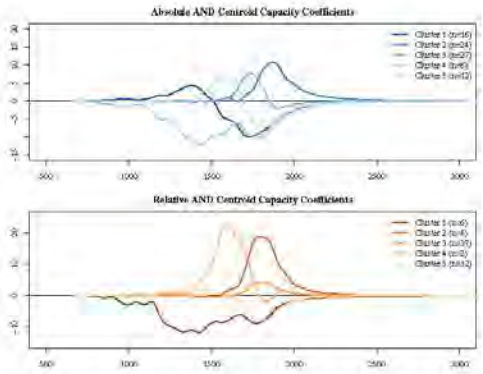

Figure 2: Representative $C_{and}(t)$ functions for each K-means cluster in both the absolute (upper) and relative (lower) fPCA-reduced capacity spaces. Centroid $C_{and}(t)$ functions determined by the linear combination of the centroid scores and the fPCs for each space.

(the first five days if a participant trained longer). Distance between vectors, D, was estimated with the Euclidean metric.

A heatmap of D is shown in Figure 3, with the rows ordered according to the hierarchical clustering results. Agglomerative clustering on D was performed with Ward's minimum variance method, minimizing total within-cluster variance (Ward, 1963).

Figure 4 depicts the dendrogram resulting from the hierarchical clustering analysis. It is immediately obvious that there is a clear division in fPCA score space between the data from the STST and AND conditions. The red bounding boxes illustrate a cut tree with four groupings. Note that increasing the number of groups in the cut tree further divided the $C_{and}(t)$ half of the dendrogram, leaving the $C_{stst}(t)$ half of the dendrogram clustered into a single group. One group contains all the $C_{stst}(t)$ fPCA vectors, indicating that all participants exhibited similar changes in $C_{stst}(t)$ over the course of learning. The heatmap in Figure 3 illustrating the values of D confirms that all $C_{stst}(t)$ fPCA scores were highly similar.

The AND scores were split into three groups. Subject 9 separates early into her own cluster (confirmed by pairwise D values at the high end of the range), reflecting a unique pattern of AND learning different from all other participants. As illustrated in Figure 1, Subject 9 exhibited a unique combination of an increasing, strictly super absolute capacity learning curve with a U-shaped relative capacity learning pattern, resulting from a strong speed-accuracy trade-off.

The second AND cluster in the middle of the dendrogram contains the majority of the absolute $C_{and}(t)$ results. This indicates that in the fPCA-reduced space, most participants exhibited similar learning-based changes in their capacity coefficients. Looking back at Figure 1, we can see the some of

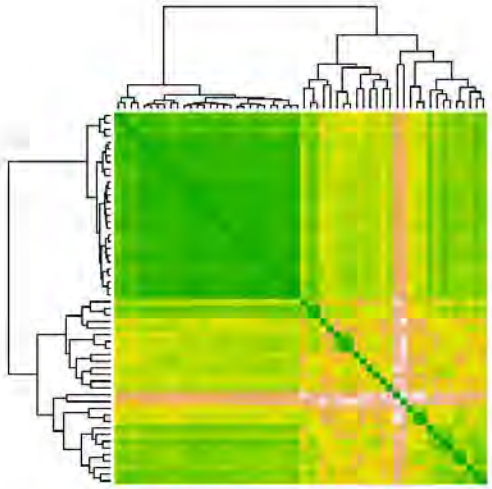

Figure 3: Heatmap of Euclidean distances, D, in the 20D fPCA scores space. The columns are ordered according to the dendrogram depicted in the margins from the Wald hierarchical clustering. Green coloring gives smaller distances, while white gives the largest distances.

the similarities in their profiles based on the K-means clustering of the individual curves. The colors in Figure 1 illustrate that participants in this cluster have capacity coefficients landing in all 5 clusters. That is, their learning trajectories move through all the average function shapes illustrated in the absolute capacity centroid AND coefficients of Figure 2. This cluster also contains a subgroup of relative fPCA score vectors, which cluster with each other before clustering with the AND vectors.

Similarly, the final AND cluster contained mostly relative capacity fPCA score vectors. Referring back to the K-means color coding for the relative capacity coefficients in Figure 1, this cluster represents those participants whose learning trajectories, measured by relative capacity, contain at least one function falling into all the capacity coefficient shapes illustrated by the relative centroid functions in the lower part of Figure 2. Again, there is a small subgroup of absolute capacity vectors that clustered into a similar part of 20D fPCA-reduced space.

We note that the small subgroup of relative (absolute) capacity scores clustering with the majority of the absolute (relative) capacity scores doesn't mean the original capacity coefficients were the same between these two groups. This is because the fPCA scores were derived separately on the two types of capacity measures, and the weights in the fPCA-reduced space refer to different fPCs. What is important is that the separation of these small subgroups from the rest of

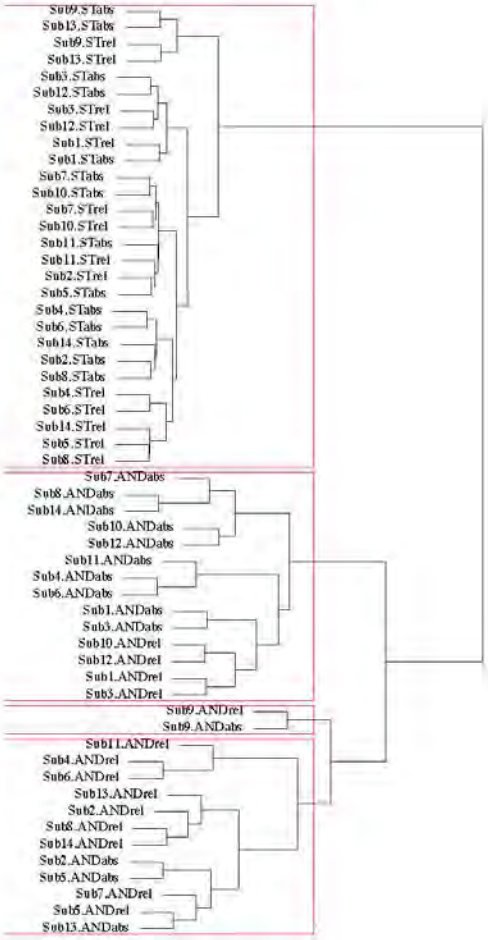

Figure 4: Dendrogram visualization of hierarchical clustering on 20D fPCA score vector space. The red boundaries indicate the cut tree segmentation into group groupings.

the data in their respective conditions implies that whether measured in absolute terms or relative terms, configural learning can be supported by two different learning trends (three if you count the trend of Subject 9). With few exceptions in this data set, the subgroups consist of individual participants whose absolute and relative capacity measures clustered into similar portions of the fPCA-reduced space. But further analysis is needed to understand how this relates to similarity between the fPCs and other functional measures.

Discussion

In this paper we have demonstrated the use of clustering techniques to explore individual differences in configural learning. The capacity coefficient gives a model-based measure of how people are using the information sources together without making specific assumptions about the RT distributions. Although the raw functions may be unwieldy for exploring sub-groups of participants, fPCA can be used to capture the important variation across capacity functions. We then used standard clustering techniques to examine different performance patterns. The cluster memberships attained with these methods can either be used for additional exploratory analysis or for further comparisons with other types of data (e.g., clinical diagnosis or working memory capacity). Importantly, clustering and other statistical learning approaches can provide principled methods for finding generalizable patterns or trends in individual data without losing the characteristics in the individual participant data, which can be particularly challenging for functional or time series data.

In previous applications of the capacity coefficient, analysis had been confined to either qualitative, verbal descriptions of different patterns across capacity functions (e.g., Group A tends to have higher capacity than Group B...) or analysis of the capacity information aggregated across time using the statistic from Houpt and Townsend (2012). The approach presented in this paper allowed us to objectively identify the different patterns of configural learning across participants using the full functional information from the capacity coefficient. From this we are able to conclude that configural learning requires at least five unique $C(t)$ function shapes to describe all the observed stages of learning captured in $C(t)$ functions. Each participant fell into one of three learning patterns, identified by the hierarchical clustering. So while all participants unitized the objects in the task and showed overall increases in RT and improvements in efficiency, there were three different trajectories through capacity coefficient functional space to get to that same trained end state. But this analysis also revealed that multiple ways of measuring capacity (absolute and relative) were needed to identify these learning patterns.

An alternative approach to extracting summary statistics from individual participants' RTs would be to fit a model and then compare model parameters (cf. Eidels, Donkin, Brown, & Heathcote, 2010). The downside to the model fitting approach is that it relies on a number of assumptions about how the RTs are generated that are ancillary to the analysis of the effect of workload (and hence the degree of configural learning). In our current approach, as with most approaches, ancillary assumptions are necessary (e.g., Euclidean distances for the clustering metrics), however these measurement assumptions are far less constraining with respect to the potential underlying processes than direct assumptions about the RT distributions. Clustering and other statistical learning methods applied to the full functional $C(t)$ data enables principled, quantified individual differences analysis with minimal assumptions about the best parametric model for capturing the underlying cognitive processes.

Acknowledgments

This work was supported by AFOSR Grant FA9550-13-1-0087 to J. W. Houpt and AFOSR LRIR to L. M. Blaha.

References

Blaha, L. M. (2010). *A dynamic Hebbian-style model of configural learning* (Unpublished doctoral dissertation). Indiana University, Bloomington, Indiana.

Blaha, L. M., Busey, T. A., & Townsend, J. T. (2009, August). An lda approach to the neural correlates of configural learning. In N. A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31st annual conference of the cognitive science society.* Austin, TX: Cognitive Science Society.

Burns, D. M., Houpt, J. W., Townsend, J. T., & Endres, M. J. (2013). Functional principal components analysis of workload capacity functions. *Behavior Research Methods, 45*, 1048-1057. doi: 10.3758/s13428-013-0333-2

Eidels, A., Donkin, C., Brown, S. D., & Heathcote, A. (2010). Converging measures of workload capacity. *Psychonomic Bulletin & Review, 17*, 763-771.

Goldstone, R. L. (2000). Unitization during category learning. *Journal of Experimental Psychology: Human Perception and Performance, 26*, 86-112.

Houpt, J. W., Blaha, L. M., McIntire, J. P., Havig, P. R., & Townsend, J. T. (2013). Systems Factorial Technology with R. *Behavior Research Methods, 46*, 307-330. doi: 10.3758/s13248-013-0377-3

Houpt, J. W., & Townsend, J. T. (2012). Statistical measures for workload capacity analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Ramsay, J. O., & Silverman, B. W. (2005). *Functional data analysis* (2nd ed.). New York: Springer.

Townsend, J. T., & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Townsend, J. T., & Wenger, M. J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003-1035.

Ward, J. H., Jr. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association, 58*, 236-244.

List of Acronyms

ACT-R	Adaptive Control of Thought - Rational
DFP	Double Factorial Paradigm
LBA	Linear Ballistic Accumulator
mMATB	Modified Multi-Attribute Task Battery
$P2P^2$	Points to Pixels Pipeline
pdf	Portable Document Format
sft	R for statistical computing package implementing systems factorial technology
SFT	Systems Factorial Technology
SIMCog-JS	Simplified Interfacing for Modeling Cognition - JavaScript